IM&F시리즈 14

금융공학 IV

Monte Carlo Methods for Finance and Economics

최병선 지음

김구재단

머릿글

최근 들어 본저자가 자주 하는 말은 "선형(linear)의 시대는 갔다. 이제는 비선형(nonlinear)의 시대이다. 나는 선형시대의 마지막 수혜자다"라는 것이다. 본저자가 대학원까지 받은 교육이 추구하던 바는 우리가 이해하고자 하는 현상을 선형모형으로 나타내고 이를 해석적으로 풀어서 소위 닫힌해(closed solution)를 구하는 것이었다. 닫힌해를 구한다는 것, 즉 해석해를 구한다는 것은 고등수학을 사용해서 문제를 푸는 것이다. 선형모형이라는 것은 우리가 이해하고자 하는 현상을 근사적으로 나타내는 것이므로 원래 문제의 관점에서 보면 닫힌해는 근사해이다.

컴퓨터과학과 공학의 경이로운 발전은 이러한 사고 패러다임을 바꾸어가고 있다. 이해하고자 하는 현상을 선형모형이 아닌 비선형모형으로 나타내고, 이 비선형문제를 해석적으로 푸는 것이 아니라 과학적컴퓨팅(scientific computing)을 통해서 해결한다. 우선 비선형모형은 선형모형을 일반화한 것이니 잘만 사용하면 선형모형보다 훨씬 더 이해하고자 하는 현상을 잘 나타낼 수 있다. 또한, 급격한 컴퓨터과학/공학의 발전은 그러한 비선형문제를 좀 더 정밀하고 효율적으로 풀 수 있도록 한다. 최근 이러한 경향을 잘 나타내는 단어들로는 빅데이터(big data), 네트워크과학과 응용, 구글의 페이지랭크(Page Rank), 알파고(Alpha Go), 알고리즘 트레이딩(algorithmic trading), DSGE모형(dynamic stochastic general equilibrium model) 등이 있다.

기존 수학만을 사용해서 비선형문제를 해결하는 데는 한계가 있으므로 컴퓨터의 도움을 필요로 한다. 물론 수치해석기법을 적용해서 비선형문제를 풀 수도 있으나, 수치해석기법을 적용할 수 없는 비선형문제가 많이 있고 또 수치해석기법을 적용하는 것보다는 시뮬레이션기법을 사용하는 것이 훨씬 더 효율적인 경우가 많다. 본서에서는 시뮬레이션기법의 기초를 소개한다. 본저자는 지난 20여년 동안 학생들에게 앞으로는 금융현상이나 경제현상을 이해하고 대처하는데 과학적컴퓨팅을 적용해야한다고 강조하면서, 그러한 관점에서 강의하고 연구를 해오고 있다. 이제 은퇴를 얼마 남기지 않은 시점에서 그동안 작성해놓았던 강의안을 정리해서 과학적컴퓨팅에 관한 책 세 권을 쓸 예정이다. 그 중에서 첫 번째 책이 본서이다. 본서에서는

난수발생부터 시작해서 MCMC의 기초까지 그리고 확률미분방정식 등 확률모형에서 표본경로를 생성하는 방법을 다루고자 한다. 두 번째 책은 금융학과 경제학에서 사용되는 수치해석에 관한 것이고, 세 번째 책은 금융학과 경제학에서 사용되는 베이지안기법에 관한 것이다.

항상 그래왔듯이 본서를 만드는 데는 많은 사람들의 도움이 컸다. 본서는 10년 전에 써놓았던 강의안을 바탕으로 한 것이다. 이 강의안을 작성하는데 사용한 워드프로세서는 한글97이다. 우선 지난 2년 동안 이 한글97로 된 파일을 LaTeX파일로 전환하는 작업을 해준 백승헌군에게 감사한다. 김찬수군은 본서에 사용된 LaTeX스타일파일을 만들어주었고, 원고를 자세히 읽고 교정을 해주었다. 우수경양도 꼼꼼히 원고를 읽고 많은 조언을 해주었다. 이번에도 이해연사장님이 표지를 디자인해주셨다. 또한, 본서를 출간하는데 있어 꼼꼼하게 행정적인 처리를 해준 김구재단 노승원부장께 감사드린다. 이 분들의 도움없이는 아마 본저자의 은퇴전에 본서가 세상에 나오지 못했을 것이다.

아직도 구천을 떠돌고 있을 세월호 희생자의 원혼을 생각하며 ...

최병선

2016년 10월 31일

차 례

제 1 장

난수

> Nothing in Nature is random. ··· A thing appears random only through the incompleteness of our knowledge.
>
> Baruch Spinoza (1632-1677)

몬테카를로법을 적용하기 위해서는 먼저 난수(random number)를 발생시켜야 한다. 이 장에서는 난수에 대해서 설명하고자 한다. 확률변수열 $x_1, x_2, \cdots$ 가 서로 독립이고 각각 누적확률분포함수 $F(x)$를 따른다고 하자. 이 확률변수열의 실현값들을 누적확률분포함수 $F(x)$에서 생성된 난수들이라고 한다.

제1.1절 일양난수

1.1.1 의사난수

난수를 다루는데 있어서 일양확률분포(uniform probability distribution)는 가장 기본적인 확률분포이다. 다음과 같은 확률밀도함수 $f(x)$를 갖는 연속형 확률변수 x는 지지대 (a, b) 상에서 일양확률분포 $U(a, b)$를 따른다고 한다.

$$f(x) = \frac{1}{b-a} 1_{(a,b)}(x) \tag{1.1.1}$$

여기서 식 $a < b$이 성립한다. 확률변수 x가 일양확률분포 $U(a, b)$를 따르는 것을 다음과 같이 표기한다.

$$x \overset{d}{\sim} U(a, b) \tag{1.1.2}$$

확률변수 x 의 평균과 분산은 각각 다음과 같다.

$$E(x) = \frac{b+a}{2}, \quad Var(x) = \frac{[b-a]^2}{12} x \overset{d}{\sim} U(a,b) \tag{1.1.3}$$

일양확률변수로부터 생성된 난수를 일양난수 (uniform random number)라고 부른다. 즉, 일양확률분포 $U(0,1)$ 를 따르고 서로 독립인 확률변수열의 실현값들 $u_1, u_2, \cdots$ 를 지지대 (0,1) 상에서의 일양난수들 또는 단순히 난수들 (random numbers) 이라 한다. 난수 또는 무작위 (randomness) 라는 개념은 무한 개 수열에 관한 개념이다. 따라서 주어진 유한수열 $\{u_1, u_2, \cdots, u_n\}$ 이 난수들인지 여부를 판단할 수 없다.

일양난수들을 구하는 방법은 여러 가지가 있다. 첫째, 필요한 횟수만큼 적절한 주사위를 던진다. 둘째, 난수표를 이용한다. 셋째, 물리난수를 이용한다. 물리난수란 물리현상을 이용해서 생성한 난수를 뜻한다. 물리난수를 생성하기 위해서는 방사선을 이용하는 방법이 자주 사용된다. 방사원에서 방사되는 감마선이 일정한 시간구간 내에 발생하는 개수가 Poisson 확률분포를 따르는 것을 이용한다. 또한 다이오드에 의해서 발생되는 노이즈를 이용하는 것도 있다.

첫 번째 방법과 두 번째 방법은 아주 많은 난수들을 필요로 하는 시뮬레이션에 적합하지 않다. 시뮬레이션에 세 번째 방법이 유용하지만, 얻어진 샘플들이 정말 난수성을 갖는지를 알 수 없다. 또한 재현성 (repeatability) 이 없으므로 이 샘플들을 사용해서 얻은 결과를 검정하기가 어렵다. 따라서 어떠한 결정론적 방법을 사용해서, 난수로 간주되는 샘플들, 이른바 의사난수들 (pseudo-random numbers)을 생성할 필요성이 있다. 무작위성이라는 관점에서는 의사난수보다 물리난수가 뛰어나지만, 물리난수는 의사난수처럼 쉽게 생성할 수 없다. 대표적인 의사난수생성법들인 선형합동법, GFSR법과 Mersenne Twister(메르센 트위스터) 는 다음과 같은 단계들을 적용해서 난수들을 생성한다. 첫 번째 단계로 컴퓨터 상에서 초기값 U_0 를 준비한다. 두 번째 단계로 어떤 함수 f 를 사용해서 점화식 $U_{n+1} = f(U_n)$ 을 이용해서 $U_1, U_2, \cdots$ 을 발생시킨다. 여기서 유의할 점은 컴퓨터의 메모리가 유한이므로 발생된 난수들 $\{U_n\}$ 이 취할 수 있는 상태들의 개수도 유한이라는 것이다. 따라서, 이렇게 발생시킨 난수들은 주기성을 갖는다. 몬테카를로법에서는 이 주기가 긴 것이 좋다. 난수를 생성하는 각 모형에서 주기의 이론적 최대값을 최대주기라 부른다.

의사난수란 결정론적인 방법에 의해 생성되는 지지대 $(1,0)$ 상의 수열로서, 근사적으로 난수성을 갖는 것을 의미한다. 초창기 의사난수생성기 (psuedo-random number generator) 로는 1946년 von Neumann이 제시한 중간제곱생성기 (the middle square generator), 1950

년대에 사용하던 Fibonacci생성기 등이 있다. 오늘날에는 이들보다 뛰어난 의사난수생성기인 Lehmer생성기, 즉 선형합동생성기(linear congruential generator: LCG)가 널리 사용되고 있다. 선형합동생성기로 의사난수들을 생성하기 위해서, 먼저 정해진 자연수들 a, c그리고 M에 의해 정의되는 다음 점화식을 사용해서 수열 $\{U_n \mid n = 1, 2, \cdots\}$을 생성한다.

$$U_{n+1} = aU_n + c \pmod{M} \tag{1.1.4}$$

여기서 a를 승수(multiplier), c를 시프트(shift), M을 모듈로(modulo)라 하고, U_0를 초기값(initial value) 또는 씨앗(seed)이라 부른다. 선형합동생성기는 집합 $\{0, 1, \cdots, M-1\}$에서 무작위적으로 수열을 생성한다. 만약 이 수열의 주기가 $M-1$이면, 이 난수생성기는 완전주기(full period)를 갖는다고 한다. 만약 $c \neq 0$이면, 이를 혼합선형합동생성기(mixed LCG)라 부른다. 만약 $c = 0$이면, 이를 승산선형합동생성기(multiplicative LCG)라 부른다.

승산선형합동생성기는 다음 식을 만족한다.

$$U_n = U_0\, a^n \pmod{M} \tag{1.1.5}$$

승산선형합동생성기에서 모듈로 M이 소수라면, $a^1, a^2, \cdots, a^{M-1}$이 서로 다른 승수 a가 반드시 존재한다. 이 경우에 집합 $\left\{a^j \pmod{M} \mid j = 1, 2, \cdots, M-1\right\}$의 원소들의 개수는 $M-1$이고, 이러한 a를 M의 원시근(primitive root)이라 부른다. 이에 해당하는 승산선형합동생성기는 최대주기가 $M-1$이다. 얼마든지 큰 소수가 존재하므로, 승산선형합동법을 사용해서 주기가 아주 긴 의사난수들을 생성할 수 있다. 그러나, 현실에서는 컴퓨터하드웨어상에서 제한이 따른다. 컴퓨터 내에서는 각 기계의 고유한 단어길이(word length) 단위로 계산이 시행되므로, 단어길이를 초과하는 계산의 효율은 나쁘다. 따라서, 한 단어가 32비트인 컴퓨터에서는, 2^{32}보다 작으면서도 2^{32}에 가까운 소수를 사용하면, 고효율로 긴 주기를 갖는 의사난수들을 얻을 수 있다. 예를 들어, Mersenne 수 $2^{31} - 1 = 2,147,483,647$를 모듈로로 하는 승산선형합동생성기를 사용하면, 주기가 $2^{31} - 1$인 일양난수열을 생성할 수 있다.

예제 1.1.1 대표적인 선형합동생성기의 하나는 Apple CarbonLib에서 사용하고 있고 또한 MATLAB에서 버전 5 전까지 사용했던 것으로, M=2^{31}−1이고 a=7^5인 승산선형합동생성기이다. 이 승산선형합동생성기는 완전주기 $2^{31} - 2 \approx 2.15 \cdot 10^9$를 갖는다. 이 승산선형합동생성기로부터 난수들을 생성하기 위해서, 다음 MATLAB프로그램 ParkMillerCongruential-Method101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: ParkMillerCongruentialMethod101.m
%  S. K. Park and K. W. Miler (1988) Communications of the ACM,
%             vol. 31, pp. 1192-1201.
%  Programmed by CBS
%  Also, refer to randmcg.m by Cleve Moler
%------------------------------------------------------------------------
clear all, close all
format long
n = 20000
a = double(7^5), M = double(2^31-1)
U(1) = 1
for ii = 2:n
    U(ii) = mod(16807*U(ii-1),2147483647);
end
[Nheight,Ucenter] = hist(U,20)
bar(Ucenter,Nheight,0.5,'y')
set(gca,'fontsize',11,'fontweigh','bold','xlim',[-0.01 2.15]*10^9)
hold on
plot(10^9*[-0.1 2.15],[1000,1000],'r-','linewidth',2)
hold off
saveas(gcf,'ParkMillerCongruentialMethod101','epsc')
save('ParkMillerCongruentialMethod101.txt','U')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 이 승산선형합동생성기로부터 숫자들 20000개 생성해서 벡터 U에 저장한다. 이 숫자들의 히스토그램이 그림 1.1.1에 그려져 있다. 그림 1.1.1에서 알 수 있듯이, 이 숫자들은 일양확률분포에서 발생한 난수들이라 할 수 있다. ■

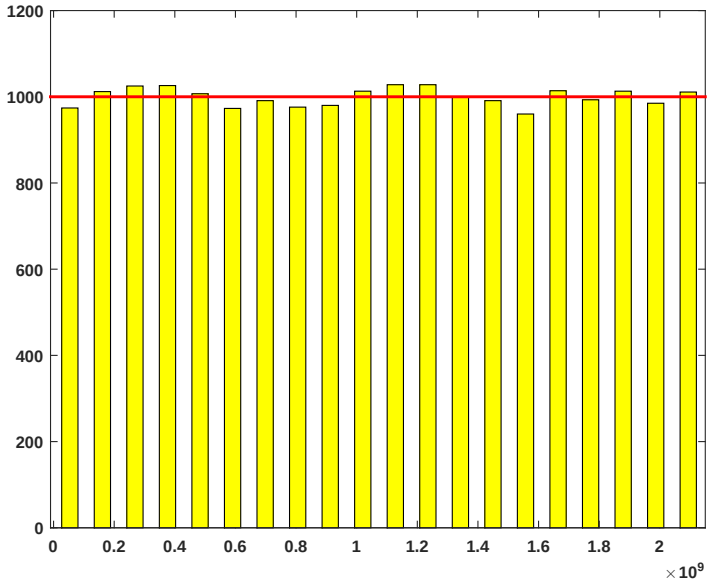


그림 1.1.1. 승산선형합동생성기

좋은 난수들을 발생시키는 난수생성기가 가져야 할 덕목은 다음과 같다. 첫째, 발생된 난수들의 질(quality)이 뛰어나야 한다. 즉, 적절한 난수성검정(randomness test)을 통과해야

하고, 같은 난수들이 다시 발생되는 주기가 상당히 길어야 한다. 둘째, 효율성(efficiency)이 있어야 한다. 즉, 난수발생에 시간이 짧게 걸리고 사용되는 메모리가 작아야 한다. 셋째, 재현성이 있어야 한다. 즉, 난수들을 바탕으로 한 실험을 재현할 수 있도록, 난수발생법은 씨앗을 이용해야 한다. 넷째, 이식가능성(portability)이 있어야 한다. 즉, 난수발생법을 다른 시스템에서 실장(implementation)해도, 같은 결과를 생산해야 한다. 다섯째, 간결성(simplicity)을 지녀야 한다. 즉, 난수생성기는 실장하기 쉬워야 한다.

잘못 발생시킨 난수들을 바탕으로 한 시뮬레이션 결과를 믿을 수 없는 것은 당연하다. 본저자가 처음 발생시켜본 난수생성기가 그러했다. 1975년 서울대학교는 당시로서는 최신 범용컴퓨터인 IBM System/360을 차관으로 도입했다. 본저자의 학사학위논문 주제는 누적확률분포함수들 사이 관계를 해석적으로 규명하는 것이었고, 그 결과를 시뮬레이션을 통해서 예증하였다. 이때, 이 시스템과 함께 도입된 사이언티픽서브루틴패키지(the Scientific Subroutine Package)에 포함된 난수생성기인 RANDU를 사용해서 난수들을 발생시켰다. 당시 본저자는 알지 못했지만 훗날 이 난수생성기에 결정적 하자가 있다는 사실을 알게 되었다.

예제 1.1.2 IBM System/360이 채택한 선형합동생성기는 다음과 같다.

$$U_{n+1} = 65539U_n \pmod{2^{31}}, \quad (n = 0, 1, \cdots) \tag{1}$$

여기서 초기값 U_0는 홀수이다. 다음 식이 성립한다.

$$65539 = 2^{16} + 3 \tag{2}$$

따라서, 다음 식들이 성립한다.

$$U_{n+1} = [\,6 \cdot 2^{16} + 18\,]\,U_n \tag{3}$$

$$U_{n+2} = [\,2^{16} + 3\,]^2 U_n = [\,6 \cdot 2^{16} + 9\,]\,U_n \tag{4}$$

식 (3)과 식 (4)에서 알 수 있듯이, 다음 식이 성립한다.

$$U_{n+2} - 6\,U_{n+1} + 9\,U_n = 0 \pmod{2^{31}}, \quad (n = 0, 1, \cdots) \tag{5}$$

즉, 이 선형합동생성기에 의해서 발생된 이웃하는 3개 숫자들은 $\pmod{2^{31}}$ 연산 하에서 선형관계를 갖는다. 따라서, RANDU는 좋은 난수생성기가 아니다.

Marsaglia [29]는 선형합동생성기에 의해서 생성한 난수들로 이루어진 점 $[U_i, U_{i+1}, U_{i+2}]$를 3차원 공간에 표시하면, 모든 점들은 유한 개의 2차원 평면들 중 하나 위에 위치하게 된다는 것을 발견하였다. 그의 이론에 의하면, 모듈로가 $M = 2^{31}$ 인 선형합동생성기가 생성할 수 있는 2차원 평면들의 최대수가 1290개지만 RANDU에 의해 발생된 난수들은 15개 평면들만을 구성한다.

승산선형합동생성기 RANDU로부터 난수들을 생성하고 이들의 3차원 그래프를 그리기 위해서, 다음 MATLAB프로그램 Ah_RANDU101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: Ah_RANDU101.m
%  Problem of IBM SYSTEM/360 Random Number Generator RANDU
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
format long
n = 10000
a = double(65539), M = double(2^31)
dum = 9 - 6*a + a^2
resid = mod(dum,M)
% RANDU
U(1) = 1
for ii = 2:n
    U(ii) = mod(65539*U(ii-1),2147483648);
end
% Plotting
plot3(U(1:end-2),U(2:end-1),U(3:end),'k.')
set(gca,'fontsize',11,'fontweigh','bold')
axis([ 0 M 0 M 0 M ])
saveas(gcf,'Ah_RANDU101','epsc')
save('Ah_RANDU101.txt','U')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 다음 식이 성립함을 확인할 수 있다.

$$9 - 6 \cdot 65539 + 65539^2 = 2 \cdot 2^{31} \tag{6}$$

또한, RANDU로부터 숫자들 10000개 생성해서 벡터 U에 저장되고, 이 숫자들의 3차원 산점도 $\{[U_i, U_{i+1}, U_{i+2}]\}$가 그림 1.1.2에 그려진다. 그림 1.1.2에서 알 수 있듯이, 이 3차원 점들은 15개 2차원 평면들 위에만 존재한다. 따라서, 이 숫자들을 난수들로 해서 시뮬레이션을 하면, 심각한 오류가 발생할 수 있다. 예를 들어, 3차원 공간에서 적분하는 문제에서 전혀 엉뚱한 결과를 얻을 수도 있다. ■

다음 단계로 식 (1.1.4)에 의해 생성된 U_{n+1}을 모듈로 M으로 나눈다. 즉, 다음 식을

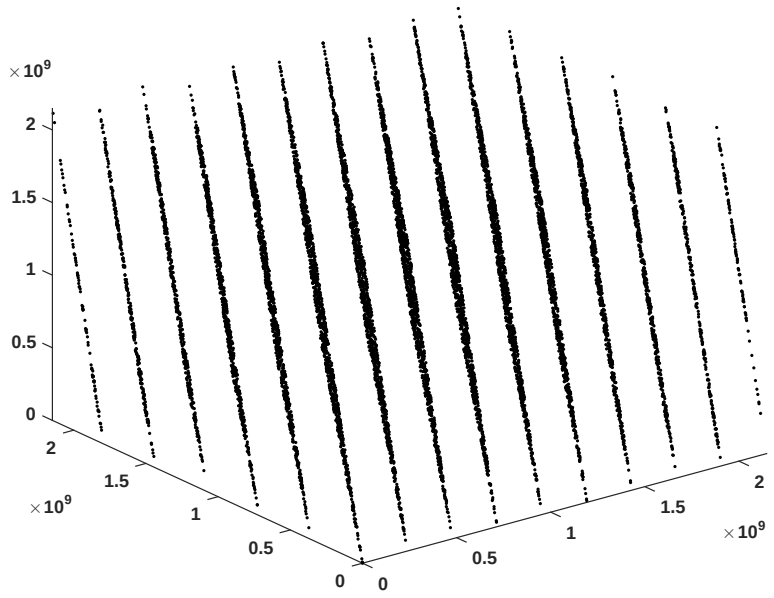


그림 1.1.2. 난수생성기 RANDU

만족하는 수열 $\{u_n \,|\, n = 1, 2, \cdots\}$을 발생시킨다.

$$u_{n+1} = \frac{U_{n+1}}{M} \tag{1.1.6}$$

따라서, u_{n+1}은 구간 $(0, 1)$에 속한다. 만약 상수들 a, c 그리고 M을 잘 선택하면, 좋은 의사난수들 $\{u_n\}$을 생성할 수 있다.

1.1.2 일양난수와 MATLAB

앞에서도 설명했듯이, 난수 생성의 시작은 일양난수를 생성하는 것이다. 일양난수를 생성하는 MATLAB함수는 rand이다. MATLAB에서 버전 4까지는 예제 1.1.1에서 설명한 선형합동생성기를 사용하였다. MATLAB에서는 이 프로그램을 mcg16807이라 부르며, 이 rand의 씨앗을 지정하기 위해서는 MATLAB명령문들 `rand('seed',sd)` 또는 `rng(sd,'v4')`를 사용한다. 여기서 sd는 0과 $2^{32} - 1$사이에 존재하는 정수이다. 이 선형합동생성기는 완전주기 $2^{31} - 2$를 가지며, 생성되는 의사난수의 범위는 $\left[\frac{1}{2^{31}-1}, 1 - \frac{1}{2^{31}-1}\right]$이다. Intel486을 사용하던 시대에 이 주기는 아주 큰 것이었다. 그러나, 최근 컴퓨터로는 이보다 훨씬 주기가 큰 난수생성기를 사용할 수 있다.

예제 1.1.3 선형합동생성기 mcg16807를 살펴보기 위해서, 다음 MATLAB프로그램 RANDseedLCG101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: RANDseedLCG101.m
%  Seed of MATLAB function RAND by LCG
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all, format long
diary RANDseedLCG101.txt
sd1 = (1001^2+1)/2
rand('seed',sd1)
y1 = rand
rng(sd1,'v4')
z1 = rand
diary off
% End of program
% ------------------------------------------------------------------
```

MATLAB명령문 'sd1 = (10012+1)/2'는 501001을 출력변수 sd1에 저장한다. MATLAB명령문 'rand('seed',sd1)'는 난수발생기 mcg16807에 씨앗을 sd1으로 하라는 것이다. MATLAB명령문 'y1 = rand'는 이 씨앗을 난수발생기 mcg16807에 적용해서 발생시킨 일양난수를 y1에 저장한다. MATLAB명령문 'rng(sd1,'v4')'도 난수발생기 mcg16807에 씨앗을 sd1으로 하라는 것이다. 여기서 v4는 MATLAB 버전 4를 의미한다. MATLAB명령문 'z1 = rand'는 이 씨앗을 난수발생기 mcg16807에 적용해서 발생시킨 일양난수를 z1에 저장한다.

이 MATLAB프로그램을 실행한 결과는 다음과 같다.

```
sd1 =  501001
y1 =   0.146129436858990
z1 =   0.146129436858990
```

이 결과물에서 알 수 있듯이, 변수들 y1과 z1은 동일한 값을 갖는다. ■

경제학이나 금융공학에서 몬테카를로법을 적용하는 문제는 점차 규모가 커지고 있다. 따라서 모듈로가 $2^{31}-1$인 승산선형합동법으로 생성한 의사난수들의 주기가 충분히 길지 않다. 이에 좀 더 주기가 긴 난수열을 발생하는 방법으로 GFSR법(Generalized Feedback Shift-Register method)을 사용한다. 이 난수생성법의 특징은 주기가 컴퓨터의 단어길이에 제약을 받지 않고 일양난수를 생성하는 것이다. 예를 들어, 가로로 나열한 32개의 비트열로 1단어를 나타내고, 521단어들을 세로로 쌓으면, 32×521 비트로 이루어진 직사각형이 만들어진다. 이 직사각형을 세로방향으로 자르면 각각 521비트인 종이조각 32매가 만들어 진다. 이 종이조각 1매가 취할 수 있는 상태들의 개수는 모든 원소들이 0인 경우를 제외하고 $2^{521}-1$이 된다. GFSR법에서는 Galois체 GF(2)상에서 원시다항식을 사용하여, 최대주기의 열을 얻는다. 이에 대한 자세한 내용은 Miyazawa & Fushimi [35]를 참조하라.

GFSR법에서는 전체 32×521 비트의 메모리를 사용하면서 주기가 $2^{521} - 1$ 인 난수열을 생성한다. 이것은 32매의 종이조각들이 동일한 상태집합에서 발생된 난수들을 기록하기 때문이다. 따라서, 만약 32×521 비트 전체에서 상태의 최대주기열을 만드는 것이 가능하다면, 더욱 긴 주기의 의사난수들을 얻을 수 있다. Matsumoto & Nishimura [34]는 GSFR법보다도 상당히 긴 주기를 실현하는 Mersenne트위스터를 제안했다. 가로로 나열한 32개의 비트열로 1단어를 나타내고, 624단어를 세로로 쌓아올려, 가장 위인 제624단 째에 관해서는 왼쪽 끝의 1비트만 사용하는 것으로 한다. 이 모서리를 빠뜨린 상자가 취할 수 있는 상태의 개수는 모든 원소들이 0인 경우를 제외하고 $2^{19937} - 1$ 이다. 이 $2^{19937} - 1$ 은 Mersenne수이며, 이것이 Mersenne트위스터라는 명칭의 유래이다. Matsumoto & Nishimura [34]는 유한이진체(finite binary field)에서 정의되는 행렬의 선형반복성을 이용해서 점화식 $u_{n+1} = f(u_n)$ 을 구성하는 함수 f 를 선택함으로써 아주 큰 주기 $2^{19937} - 1$ 를 갖는 의사난수생성기를 만들었다.

MathWorks사는 MATLAB 버전 5부터 선형합동생성기가 아닌 다른 난수생성기를 채택하였다. 이 난수생성기는 Marsaglia & Zaman [33]이 제시한 방법을 바탕으로 한 것으로서 Fibonacci생성기에 '`substract-with-borrow`'단계를 포함한 것이다. 이 난수생성기를 SWB생성기라 부르자. 선형합동생성기에서는 자연수 $k\,(= 0, 1, \cdots, M-1)$ 를 발생시키고 이를 모듈로 M 으로 나누어 지지대 $(0, 1)$ 에서 일양난수들을 발생시킨다. 따라서, 이렇게 발생된 난수의 형태는 k/M 이다. 그러나, SWB생성기는 변동소수점값(floating point value)을 직접 계산한다. 선형합동계산기에서 한 개의 씨앗(seed)를 사용하는 반면에, SWB생성기는 35개 원소들로 구성된 초기벡터를 필요로 한다. 이 중 첫 $2^6 = 32$ 개 원소들은 지지대 $(0, 1)$ 에서 발생된 일양난수들이다. 이 난수들은 적절한 선형합동난수기에서 발생된 것들이다. 또한, 나머지 세 원소들은 상태를 나타내는 숫자 i, '`borrow`'에 해당하는 숫자 b 그리고 초기상태를 결정하는 확률정수(random integer) j 이다. 여기서 숫자 i 는 $'i \pmod{32}'$ 를 의미하므로 집합 $\{0, 1, \cdots, 31\}$ 에 속한다. 이미 발생된 일양난수들 $z_0, z_1, \cdots, z_{31}$ 로부터 다음 식을 이용해서 새로운 수 z_i^0 를 발생시킨다.

$$z_i^0 = z_{i+20} - z_{i+5} - b \tag{1.1.7}$$

여기서 b는 바로 전 단계에서 결정된 값이다. 만약 식 $z_i^0 \geq 0$ 이 성립하면, 다음과 같이 새로운 난수 z_i 와 다음 단계를 위한 b를 결정한다.

$$z_i = z_i^0, \quad b = 0 \tag{1.1.8}$$

만약 식 $z_i^0 < 0$이 성립하면, 다음과 같이 새로운 난수 z_i와 다음 단계를 위한 b를 결정한다.

$$z_i = z_i^0 + 1, \quad b = 2^{-53} \tag{1.1.9}$$

이 SWB생성기의 주기는 2^{1430}이다. MATLAB에서는 초기상태를 나타내는 j와 배타적 OR(exclusive OR)를 이용해서 이 SWB생성기의 질을 향상시킨 수정SWB생성기를 사용한다. 이 수정SWB생성기 rand의 주기는 $2^{1492} \approx 1.37 \cdot 10^{449}$이며, 생성되는 의사난수의 범위는 $[2^{-53}, 1 - 2^{-53}]$이다. 수정SWB생성기 rand에서는 곱하기나 나누기를 사용하지 않는다는 점에 유의하라. MATLAB 버전 5부터 7.3까지는 이 수정SWB생성기를 사용한 rand가 디폴트이다. MATLAB에 `'rng(sd,'v5uniform')'`을 사용한다. 여기서 sd는 0과 $2^{32} - 1$사이에 존재하는 정수이다.

예제 1.1.4 선형합동생성기 swb2712를 살펴보기 위해서, 다음 MATLAB프로그램 RANDseedSWB101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: RANDseedSWB101.m
%  Seed of MATLAB function RAND by Modified SWB
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all, format long
diary RANDseedSWB101.txt
sd1 = (1001^2+1)/2
rand('state',sd1)
y1 = rand
rng(sd1,'v5uniform')
z1 = rand
diary off
% End of program
% ------------------------------------------------------------------------
```

MATLAB명령문 `'sd1 = (10012+1)/2'`는 501001을 출력변수 sd1에 저장한다. MATLAB명령문 `'rand('state',sd1)'`는 난수발생기 swb2712에 씨앗을 sd1으로 하라는 것이다. MATLAB명령문 `'y1 = rand'`는 이 씨앗을 난수발생기 swb2712에 적용해서 발생시킨 일양난수를 y1에 저장한다. MATLAB명령문 `'rng(sd1,'v5uniform')'`도 난수발생기 swb2712에 씨앗을 sd1으로 하라는 것이다. 여기서 v5는 MATLAB 버전 5를 의미한다. MATLAB명령문 `'z1 = rand'`는 이 씨앗을 난수발생기 swb2712에 적용해서 발생시킨 일양난수를 z1에 저장한다.

이 MATLAB프로그램을 실행한 결과는 다음과 같다.

```
sd1 =  501001
```

```
y1 =    0.227719620032458
z1 =    0.227719620032458
```

이 결과물에서 알 수 있듯이, 변수들 y1과 z1은 동일한 값을 갖는다. ■

MathWorks사는 MATLAB 버전 7.1부터 Mersenne트위스터를 사용한 rand를 채택하였고, MATLAB 버전 7.4부터는 이 트위스터생성기를 사용한 rand가 디폴트이다. 만약 k 비트로 구성된 워드를 사용하면, 트위스터생성기는 지지대 $[0, 2^k - 1]$에서 일양난수들을 생성한다. MathWorks사에서 제공하는 RandStream.list에 의하면, MATLAB에서 사용하는 트위스터생성기의 주기는 $2^{19937} - 1 \approx 4.31 \cdot 10^{6001}$ 이며, 생성되는 의사난수의 범위는 $[2^{-53}, 1 - 2^{-53}]$이다. MATLAB에서는 이 프로그램을 mt19937ar이라 부르며, 이 rand의 씨앗을 지정하기 위해서는 MATLAB명령문들 `'rand('twister',sd)'` 또는 `'rng(sd,'twister')'`를 사용한다. 여기서 sd는 0과 $2^{32} - 1$사이에 존재하는 정수이다. 특히, 트위스터생성기의 경우에 `'sd = 54891`가 디폴트이다.

예제 1.1.5 선형합동생성기 mt19937ar를 살펴보기 위해서, 다음 MATLAB프로그램 RANDseedTwister101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: RANDseedTwister101.m
%  Seed of MATLAB function RAND by Twister
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all, format long
diary RANDseedTwister101.txt
sd1 = (1001^2+1)/2
rand('twister',sd1);
y1 = rand
rng(sd1,'twister')
z1 = rand
diary off
% End of program
% ------------------------------------------------------------------
```

MATLAB명령문 `'sd1 = (10012+1)/2'`는 501001을 출력변수 sd1에 저장한다. MATLAB명령문 `'rand('twister',sd1)'`는 난수발생기 mt19937ar에 씨앗을 sd1으로 하라는 것이다. MATLAB명령문 `'y1 = rand'`는 이 씨앗을 난수발생기 mt19937ar에 적용해서 발생시킨 일양난수를 y1에 저장한다. MATLAB명령문 `'rng(sd1,'twister')'`도 난수발생기 mt19937ar에 씨앗을 sd1으로 하라는 것이다. MATLAB명령문 `'z1 = rand'`는 이 씨앗을 난수발생기 mt19937ar에 적용해서 발생시킨 일양난수를 z1에 저장한다.

이 MATLAB프로그램을 실행한 결과는 다음과 같다.

```
sd1 =  501001
y1 =   0.946970460124160
z1 =   0.946970460124160
```

이 결과물에서 알 수 있듯이, 변수들 y1과 z1은 동일한 값을 갖는다. ■

예제 1.1.6 MATLAB에서 사용할 수 있는 난수생성기의 종류를 살펴보기 위해서는 MATLAB명령문 `'RandStream.list'`를 실행한다. 본서를 출간하는 시점에 이 함수를 사용한 결과는 다음과 같다.

```
다음 난수 생성기 알고리즘을 사용할 수 있다.
dsfmt19937:   Mersenne소수(Mersenne Prime) 2^19937-1을 사용한 SIMD 지향 고속
              Mersenne트위스터(SFMT)
mcg16807:     승수 7^5, 모듈로 2^31-1을 사용하는 승산식 합동법(multiplicative
              congruential) 생성기
mlfg6331_64:  시차(lag) 63 및 31과 64비트를 사용하는 시차Fibonacci수열
              (multiplicative lagged Fibonacci) 생성기 (병렬스트림 지원)
mrg32k3a:     결합다중재귀적(combined multiple recursive) 생성기 (병렬 스트림 지원)
mt19937ar:    Mersenne소수 2^19937-1을 사용한 Mersenne트위스터
              (Mersenne Twister)
shr3cong:     CONG 선형합동법(CONG linear congruential) 생성기로 계산된 SHR3
              시프트-레지스터 생성기
swb2712:      시차(lag) 27과 12를 사용하는 수정된 자리 내림을 사용하는 뺄셈
              (Subtract-with-Borrow) 생성기
```

■

예제 1.1.7 MATLAB의 난수생성기 rand를 사용해서 생성한 숫자들 $\{u_i \,|\, 1, 2, \cdots, 10000\}$의 3차원 그래프 $\{[u_i, u_{i+1}, u_{i+2}]\}$가 그림 1.1.3에 그려져 있다. 그림 1.1.3을 그리기 위해서 다음 MATLAB프로그램 Oh_RAND101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: Oh_RAND101.m
%  MATLAB Random Number Generator RAND
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all, format long
n = 10000
rand('twister',5489);                    % rng('default')
u = rand(1,n);
% Plotting
```

```
plot3(u(1:end-2),u(2:end-1),u(3:end),'k.')
set(gca,'fontsize',11,'fontweigh','bold')
axis([ 0 1 0 1 0 1 ])
saveas(gcf,'Oh_RAND101','epsc')
save('Oh_RAND101.txt','u')
% End of program
% ------------------------------------------------------------------------
```

그림 1.1.3의 그래프를 어떤 방향으로 회전해도 이 점들은 3차원 공간상에 골고루 분포되어 있음을 알 수 있다. ■

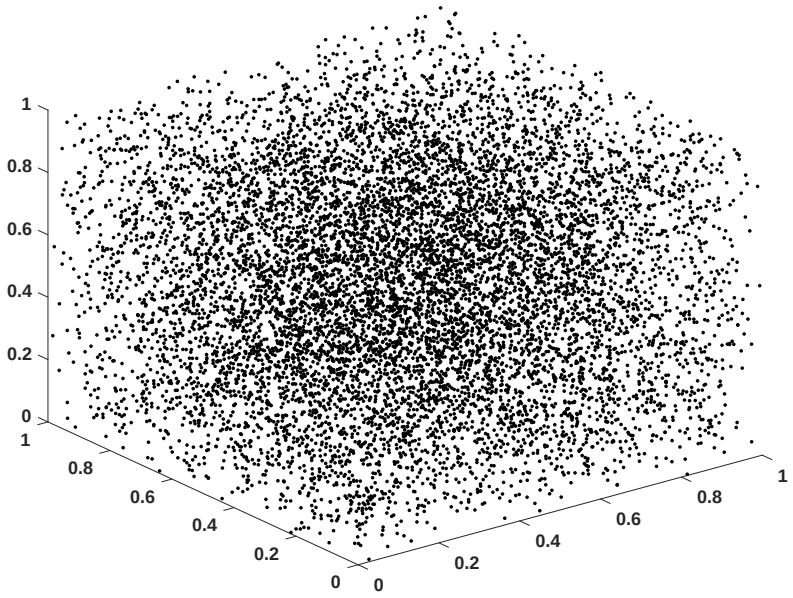


그림 1.1.3. 난수생성기 RAND

MATLAB함수 rand를 사용하는 방법을 익히기 위해서, 다음 예제를 살펴보자.

예제 1.1.8 MATLAB함수 rand를 사용해서 일양난수를 발생시키는 방법을 살펴보기 위해서, 다음 MATLAB프로그램 RANDexample101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: RANDexample101.m
%  Generating Uniform Random Numbers using MATLAB function rand
%  Programmed by CBS
%-------------------------------------------------------------------
clear all, close all, format long
rand('twister',5489)
a1 = rand
a2 = rand(2)
a3 = rand(2,3)
a4 = rand([2 3])
a5 = rand(2,3,2)
a6 = rand([2 3 2])
X = ones(3,2);
a7 = rand(size(X))
```

```
a8 = rand(1,2,'double')
a9 = rand(1,2,'single')
% End of program
% ------------------------------------------------------------------
```

첫 번째 명령문 'a1 = rand'는 일양난수 1개를 생성해서 a1에 저장한다. 두 번째 명령문 'a2 = rand(2)'는 일양난수들로 이루어진 2×2 행렬을 생성해서 a2에 저장한다. 세 번째 명령문 'a3 = rand(2,3)'는 일양난수들로 이루어진 2×3 행렬을 생성해서 a3에 저장한다. 네 번째 명령문 'a4 = rand([2 3])'는 세 번째 명령문 'a3 = rand(2,3)'와 같다. 다섯 번째 명령문 'a5 = rand(2,3,2)'는 일양난수들로 이루어진 $2 \times 3 \times 2$ 행렬을 생성해서 a5에 저장한다. 여섯 번째 명령문 'a6 = rand([2 3 2])'는 다섯 번째 명령문 'a5 = rand([2 3 2])'와 같다. 일곱 번째 명령문 'a7 = rand(size(X))'는 일양난수들로 이루어진 행렬 X와 같은 차원의 행렬을 생성해서 a7에 저장한다. 여덟 번째 명령문 'a8 = rand(1,2,'double')'은 2배정도(double precision) 일양난수들로 이루어진 1×2 행렬을 생성해서 a8에 저장한다. 아홉 번째 명령문 'a9 = rand(1,2,'single')'은 1배정도(single precision) 일양난수들로 이루어진 1×2 행렬을 생성해서 a9에 저장한다. ■

일양난수로부터 이산형 확률변수의 난수를 생성하기 위해서는 실수를 특정한 정수로 변환시키는 함수가 필요한다. 이러한 MATLAB함수들로는 ceil, floor, round 그리고 fix가 있다. 이 함수들의 사용법을 익히기 위해서 다음 예제를 살펴보자.

예제 1.1.9 MATLAB함수들 ceil, floor, round 그리고 fix의 사용법을 익히기 위해서, 다음 MATLAB프로그램 Real2Integer101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: Real2Integer101.m
%  Functions from a Real Number to an Integer
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
x = -2.5:1:2.5
cx = ceil(x)
fx = floor(x)
rx = round(x)
fixx = fix(x)
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행한 결과는 다음과 같다.

```
x    = -2.5000   -1.5000   -0.5000    0.5000    1.5000    2.5000
```

```
cx   =  -2    -1     0     1     2     3
fx   =  -3    -2    -1     0     1     2
rx   =  -3    -2    -1     1     2     3
fixx =  -2    -1     0     0     1     2
```

즉, MATLAB함수 ceil(a)는 a 이상인 정수 중에서 a에 가장 가까운 정수를 나타내고, MATLAB함수 floor(a)는 a 이하인 정수 중에서 a에 가장 가까운 정수를 나타내고, MATLAB함수 round(a)는 a를 반올림한 정수를 나타내고, MATLAB함수 fix(a)는 0쪽으로 a와 가장 가까운 정수를 나타낸다. 즉, a가 양수이면 fix(a)는 floor(a)와 같고 a가 음수이면 fix(a)는 ceil(a)와 같다. ■

지금까지 설명한 방법들을 복습하는 뜻에서, 이산형 확률분포에서 난수를 생성해보자.

예제 1.1.10 MATLAB을 사용해서 다음과 같은 이산형 확률밀도함수를 갖는 확률분포에서 난수들을 생성하자.

$$p(x=10)=\frac{1}{6},\quad p(x=11)=\frac{1}{3},\quad p(x=12)=\frac{1}{3},\quad p(x=13)=\frac{1}{6} \tag{1}$$

이 확률분포에서 20개 난수들을 생성하기 위해서, 다음 MATLAB프로그램 GeneratingDiscreteRN101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: GeneratingRN101.m
%  Generating Random Number
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all
rng((1001^2+1)/2,'twister')
n = 20
p = [ 1/6 2/6 2/6 1/6 ]
cump = cumsum(p)
u = rand(1,n)
% Method 1
tic
xran = zeros(1,n)+10*(u < cump(1));
for ii=2:4
xran = xran+(9+ii)*(cump(ii-1) <= u & u < cump(ii));
end
xran
toc
% Method 2
tic
v = floor(6*u)
yran = ones(1,n);
for jj=1:n
    switch v(jj)
```

```
        case 0
             yran(jj) = 10;
        case 5
             yran(jj) = 13;
        case 3
             yran(jj) = 12;
        case 4
             yran(jj) = 12;
        otherwise
             yran(jj) = 11;
     end
end
yran
toc
% End of program
% --------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 두 가지 방법들에 의해서 난수들을 생성한다. 두 방법들 모두 다음 난수들을 생성한다.

```
12, 12, 12, 12, 11, 11, 11, 11, 10, 11
12, 12, 12, 12, 10, 13, 10, 11, 13, 12
```

■

예제 1.1.11 R을 사용해서 다음과 같은 이산형 확률밀도함수를 갖는 확률분포에서 난수들을 생성하자.

$$p(x=10)=\frac{1}{6},\quad p(x=11)=\frac{1}{3},\quad p(x=12)=\frac{1}{3},\quad p(x=13)=\frac{1}{6} \tag{1}$$

이 확률분포에서 20개 난수들을 생성하기 위해서, 다음 R프로그램 GeneratingDiscreteRN101R.R을 실행해 보자.

```
# --------------------------------------------------------------------------
# Filename: GeneratingDiscreteRN101R.R
# Generating Random Number
# Programmed by CBS
# --------------------------------------------------------------------------
set.seed(11)
yran <-sample(x=c(10,11,12,13), size=20, replace=TRUE, prob=c(1/6,1/3,1/3,1/6))
yran
# --------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 다음 난수들을 생성한다.

```
11 11 12 11 11 10 11 11 10 11
11 12 10 10 13 12 12 11 11 12
```

■

주어진 p개 대상물들을 무작위적으로(ramdomly) 재배열하거나 이 p개 대상물들에서 무작위로 m개를 선택하는 방법들을 살펴보자. 이러한 방법들은 주어진 개체들에서 몇 개를 무작위적으로 선택하는 표본추출이나 비모수적 검정에서 유의확률을 계산할 때 이용된다.

집합 $\{1, 2, \cdots, p\}$의 원소들을 일렬로 나열하는 방법은 $p!$개이고, 이 $p!$개 순열들 중 1개를 뽑는 확률은 $1/p!$이다. 일양난수를 이용해서 이러한 일양난순열(umiform random permutation)을 생성할 수 있다. 우선, 1부터 p까지를 다음과 같은 배열 $\{x(i)\}$에 할당한다.

$$x(1) = 1,\ x(2) = 2,\ \cdots,\ x(p) = p \tag{1.1.10}$$

다음과 같은 절차를 $j = 1$부터 $p - 1$까지 반복한다. 첫째 단계로 집합 $\{j, j+1, \cdots, p\}$에서 한 숫자를 무작위로 선택한다. 이 숫자를 k라 하자. 둘째 단계 $x(j)$와 $x(k)$가 포함한 숫자들을 맞바꾼다. 이러한 과정을 끝내면, $x(1)$에 어떤 숫자 i_1이 할당될 확률은 $1/p$이고, $x(2)$에 숫자 i_1가 아닌 숫자 i_2가 나올 확률은 $1/[p-1]$이며, $x(3)$에 숫자 i_1이나 숫자 i_2가 아닌 숫자 i_3가 나올 확률은 $1/[p-2]$이다. 각 단계에서 숫자가 선택되는 사상은 과거와 독립이므로, 어떤 특정한 순열 $[i_1, i_2, \cdots, i_p]$가 선택될 확률은 $1/p!$이다. 집합 $\{j, j+1, \cdots, p\}$에서 무작위로 하나의 숫자를 선택하기 위해서는 먼저 지지대가 $[0, 1]$인 일양난수 u를 선택한 다음, $j + \lfloor u[p - j + 1] \rfloor$를 계산한다.

예제 1.1.12 일양난수를 이용해서 일양난순열을 생성하는 방법은 간단하다. 우선 서로 독립이고 지지대가 (0,1)인 일양난수열 $u_1, u_2, \cdots, u_p$를 생성하고, 이들을 다음과 같이 올림차순으로 나열한다.

$$u_{x_1} \le u_{x_2} \le \cdots \le u_{x_p} \tag{1}$$

이 $\boldsymbol{x} = [x_1, x_2, \cdots, x_p]$가 일양난순열이다. 이 방법을 사용해서 일양난순열을 생성하기 위해서 다음 MATLAB프로그램 RandomPermutation101.m을 실행해 보자.

```
% ------------------------------------------------------------------
% Filename: RandomPermutation101.m
% Generating Random Permutions 1
% w/o using MATLAB function 'randperm'
% Programmed by CBS
% ------------------------------------------------------------------
```

```
clear all, close all
n = 2                % Number of Random Permutation
p = 100
k = 11
rng((1001^2+1)/2,'twister')
for ii = 1:n
    [ srand indrand ] = sort(rand(1,p));
    x(ii,:) = indrand(1:k);
end
x
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 집합 $\{1, 2, \cdots, 100\}$에서 서로 다른 $k = 11$개 원소들을 뽑은 난순열 2개를 출력한다. 그 난순열들은 다음과 같다.

```
[99 32 40 22 34 92 91 35 6 55 3]
[23 74 75 52 70 46 19 90 12 27 65]
```

■

예제 1.1.13 R을 사용해서 일양난순열을 생성하기 위해, 다음 R프로그램 RandomPermutation101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: RandomPermutaion101R.R
# Generating Random Number
# Programmed by CBS
# ------------------------------------------------------------------------
set.seed(11)
yperm1 <-sample(x=c(1:100), size=11, replace=FALSE, prob=NULL)
yperm2 <-sample(x=c(1:100), size=11, replace=FALSE, prob=NULL)
yperm <- rbind(yperm1,yperm2)
yperm
# ------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 집합 $\{1, 2, \cdots, 100\}$에서 서로 다른 $k = 11$개 원소들을 뽑은 난순열 2개를 출력한다. 그 난순열들은 다음과 같다.

```
       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
yperm1   45   90   84   72   56   46   32   15  100    19    62
yperm2   37   35    7   47   39    2   12   38   97    30    93
```

■

제1.2절 적합성검정

생성된 난수들이 난수성을 갖는지 그리고 원하는 확률분포에서 생성된 것인지를 조사하는 것은 매우 중요한 일이다. 이러한 과정을 적합성검정(goodness-of-fit test)이라 한다.

1.2.1 히스토그램

난수성검정을 하기 위해서는 먼저 난수들의 히스토그램을 그려볼 필요가 있다. 히스토그램을 그리는 MATLAB함수는 histogram.m이다. 이 함수의 간단한 사용법을 익히기 위해서 다음 예제를 살펴보자.

예제 1.2.1 MATLAB함수 hist를 사용해서 히스토그램을 그리는 방법을 살펴보기 위해서, 다음 MATLAB프로그램 HistogramRN101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: HistogramRN101.m
%  Plotting Histograms of Random Numbers
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all
rng((1001^2+1)/2,'twister')
yran = randn(1,256);
% Histogram 1
subplot(2,2,1)
h1 = histogram(yran)
set(gca,'fontsize',11,'fontweigh','bold')
% Histogram 2
subplot(2,2,2)
nbins = 25;
h2 = histogram(yran,nbins,'Normalization','pdf')
h2.FaceColor = [0.1 0.7 0.1];
hold on
yy = -3.5:0.1:3.5;
muu=0; sigmaa=1;
f = exp(-(yy-muu).^2./(2*sigmaa^2))./(sigmaa*sqrt(2*pi));
plot(yy,f,'k','LineWidth',1.5)
set(gca,'fontsize',11,'fontweigh','bold')
hold off
% Histogram 3
subplot(2,2,3)
yran31 = randn(128,1);
yran32 = -2 + randn(256,1);
h31 = histogram(yran31);
set(gca,'fontsize',11,'fontweigh','bold')
hold on
h32 = histogram(yran32);
h31.Normalization = 'probability';
h31.BinWidth = 0.25;
h32.Normalization = 'probability';
h32.BinWidth = 0.15;
xlim([-5 2.3]);
hold off
```

```
% Histogram 3
subplot(2,2,4)
edges = [-5 -3:0.5:3 5];
h4 = histogram(yran,edges,'Normalization','probability')
set(gca,'fontsize',11,'fontweigh','bold')
h4.FaceColor = [0.5 0.5 0.5];
h4.EdgeColor = 'r';
saveas(gcf,'HistogramRN101','epsc')
save('HistogramRN101','yran')
% End of program
% ------------------------------------------------------------------------
```

씨앗을 7919로 하는 일양난수들을 100개 생성해서 벡터 yran에 저장한다.

MATLAB명령문 `'h1 = histogram(yran)'`은 등간격으로 나누어진 상자들(bins), 즉 소구간들에 벡터 yran의 원소들을 할당한 다음 각 소구간에 속한 원소들의 개수를 원소로 하는 벡터 h1을 구성하고 이 히스토그램을 그린다. 이 명령문을 수행한 결과가 그림 1.2.1의 좌측상단에 있는 그래프이다.

MATLAB명령문 `'h2 = histogram(yran,nbins,'Normalization','pdf')'`은 소구간들 개수가 nbins = 25인 히스토그램을 그린다. 이 히스토그램의 그래프가 그림 1.2.1의 우측상단에 수록되어 있다. 이 히스토그램의 면적을 1로 만들어 확률밀도함수의 추정함수로 만들기 위해서, 옵션 Normalization에 pdf를 할당했다. 따라서, 이 그래프에서 녹색 부분의 면적 1이다. 또한, 정규확률밀도함수가 흑색 실선으로 그려져 있다.

MATLAB명령문 `'hold on'`을 사용해서 MATLAB명령문 `'h31 = histogram(yran31)'`에 의한 히스토그램 h31과 MATLAB명령문 `'h32 = histogram(yran32)'`에 의한 히스토그램 h32를 겹쳐 그린 그래프가 그림 1.2.1의 좌측하단에 수록되어 있다. 이 히스토그램들에서는 Y축에 관찰점들의 개수가 아닌 상대도수를 표기하기 위해, 옵션 Normalization에 probability를 할당했다.

MATLAB명령문 `'h4 = histogram(yran,edges4,'Normalization','probability')'`에서는 옵션 edges를 사용해서 상자들(bins), 즉 소구간들의 왼쪽 점들과 마지막 소구간의 오른쪽 점을 지정한다. 또한, 옵션 FaceColor를 사용해서 히스토그램의 막대 내부 색을 지정하고, 옵션 EdgeColor를 사용해서 히스토그램의 막대 테두리 색을 지정한다. ■

1.2.2 카이제곱 검정

어떤 확률분포의 지지대가 k개 상자들(bins), 즉 소구간들로 나누어져 있는 경우, 제 i번째 소구간에 포함된 관찰값들의 개수와 이에 해당하는 기대값을 각각 Q_i와 E_i라고 하면, 카이제

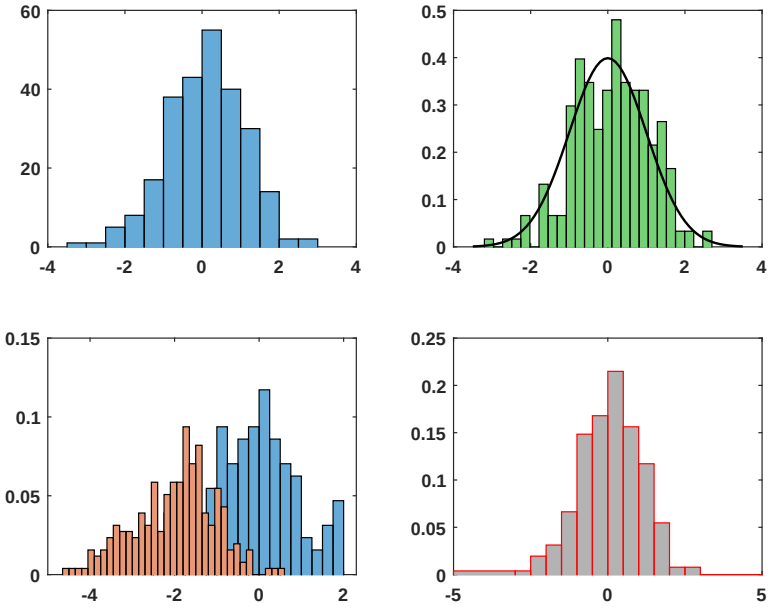


그림 1.2.1. 히스토그램

곱검정통계량은 다음과 같이 정의된다.

$$X^2 \doteq \sum_{i=1}^{k} \frac{[Q_i - E_i]^2}{E_i} \tag{1.2.1}$$

이 관찰값들이 주어진 확률분포에서 생성된 것이라는 귀무가설 하에서 다음 식이 성립한다.

$$X^2 \quad \xrightarrow{d} \quad \chi^2_{k-1-q} \tag{1.2.2}$$

여기서 q는 추정된 모수들의 개수이고, ' $\xrightarrow{d}$ ' 는 분포수렴을 의미한다. 즉, 관찰값들의 개수가 커지면, 카이제곱검정통계량 X^2는 점근적으로 자유도가 $k-1-q$인 카이제곱분포를 따른다. 귀무가설 하에서 카이제곱확률변수가 이 카이제곱통계량값 X^2보다 클 확률을 p값이라 한다. 즉, p값을 다음과 같이 정의한다.

$$p \doteq Pr(\chi^2_{k-1-q} \leq X^2) \tag{1.2.3}$$

만약 p값이 원하는 유의수준 α보다 작으면, 귀무가설을 기각한다.

카이제곱검정통계량은 우도비검정통계량을 2차까지 Taylor전개해서 구한 근사식이다. 이러한 유도과정에서 다음 조건들이 만족되는 경우에 한해서 카이제곱검정을 적용할 수 있음을 알 수 있다. 데이터는 모집단에서 무작위적으로 추출된 것이어야 하고, 관찰값들의 개수가 커야한다. 또한, 각 상자(bin), 즉 소구간에 속하는 관찰값들은 상대도수가 아닌 개수로

주어져야 하며, 각 소구간에 속하는 관찰값들의 개수가 지나치게 작지 않아야 한다.

예제 1.2.2 카이제곱통계량을 사용해서 적합성을 하기 위해서, 다음 MATLAB프로그램 ChiSquareTest101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: ChiSquareTest101.m
%  Chi-Square test of Random Numbers
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
rng((1001^2+1)/2,'twister')
n = 1000                                    % No of Observations
yran = rand(1,n);                           % Random Numbers
bin = 25                                    % No of Subintervals
[OB bins] = hist(yran,bin)                  % Observed Values
EX = n/bin*ones(1,bin)                      % Expected Values
% Plotting
bar(bins,OB,0.5,'w')
set(gca,'fontsize',11,'fontweigh','bold')
hold on
plot(bins,EX,'r-','linewidth',2)
legend('Observed','Expected','location','NW')
xlabel('\bf Bin'), ylabel('\bf Frequency')
saveas(gcf,'ChiSquareTest101','epsc')
% Chi-Square Test 1
chi2 = sum(power(OB-EX,2)./EX)
p_val = 1 - chi2cdf(chi2,bin-1)
% Chi-Square Test 2
[H,P_value,stat] = chi2gof(bins,'ctrs',bins, ...
                           'frequency',OB,'expected',EX)
save('ChiSquareTest101','chi2','p_val','stat','P_value')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 일양난수들 1000개가 생성된다. 이 일양난수들을 25개의 소구간들로 이루어진 히스토그램을 그린 것이 그림 1.2.2이다. 그림 1.2.2에서 제 i번째 막대의 높이는 관찰값들의 개수 Q_i를 나타내고 적색 별표의 높이가 기대값 E_i을 나타낸다.

첫 번째 카이제곱검정에서 카이검정통계량값이 $X^2 = 33.9500$임을 알 수 있다. MATLAB 함수 chi2cdf를 사용해서 계산한 결과, p값은 0.0856이다. 이 p값이 유의수준 0.05보다 크므로 귀무가설을 채택한다. 즉, 이 난수들이 일양확률분포에서 생성되었다는 귀무가설을 기각하지 않는다.

두 번째 카이제곱검정은 적합성 (goodness-of-fit test)를 위한 MATLAB함수 chi2gof.m을 사용한 것이다. 출력변수 H가 0이므로 귀무가설을 채택한다. 출력변수 P_value가 0.0856이므로 p값이 0.0856이다. 출력변수 stat에는 검정통계량값 chi2stat, 자유도 df, 소구간들의 경계값들을 포함한 edge, 관찰값들의 개수를 나타내는 O 그리고 기대값을 나타내는 E가

출력된다. ■

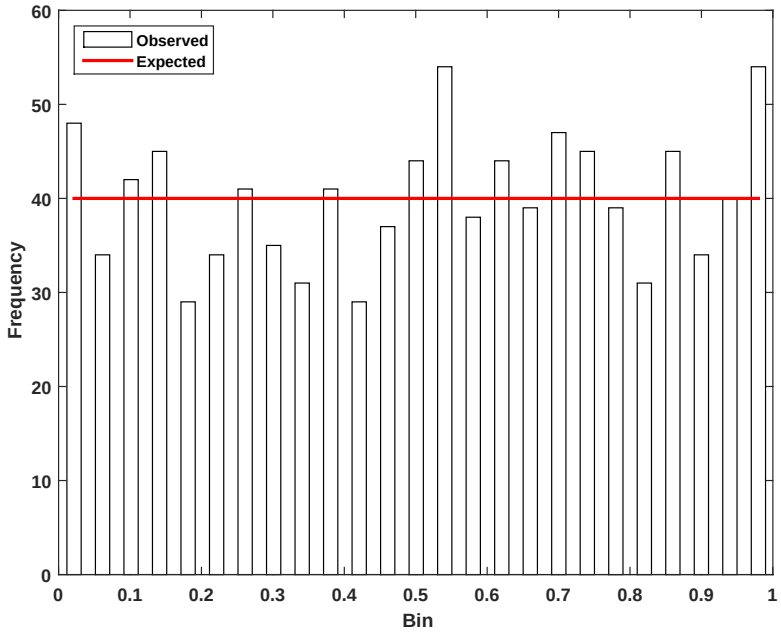


그림 1.2.2. 히스토그램과 카이제곱검정

앞에서도 언급했듯이, MATLAB함수는 cdf2gof.m을 사용하면 카이제곱검정을 할 수 있다. 이 함수의 사용법을 익히기 위해서 다음 예제를 살펴보자.

예제 1.2.3 MATLAB함수 cdf2gof.m을 사용해서 적합성을 하는 방법을 살펴보기 위해서, 다음 MATLAB프로그램 ChiSquareTest102.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: ChiSquareTest102.m
%  Examples of MATLAB function chi2gof.m
%  Programmed by MATLB
%------------------------------------------------------------------------
clear all, close all
rng((1001^2+1)/2,'twister')
x = normrnd(50,5,100,1);              % Generating Normal Random Numbers
% Test against an unspecified normal distribution with estimated parameters
[H1,p1,stat1] = chi2gof(x)
[H2,p2,stat2] = chi2gof(x,'cdf',@(z)normcdf(z,mean(x),std(x)),'nparams',2)
[H3,p3,stat3] = chi2gof(x,'cdf',{@normcdf,mean(x),std(x)})
% Test against the standard normal
x = randn(100,1);
[H4,p4,stat4] = chi2gof(x,'cdf',@normcdf)
% Test against the standard uniform
x = rand(100,1);
n = length(x);
edges = linspace(0,1,11);
expectedCounts = n*diff(edges);
[H5,p5,stat5] = chi2gof(x,'edges',edges,'expected',expectedCounts)
% Test against the Poisson dist by specifying observed and expected counts:
bins = 0:5;
```

```
obsCounts = [6 16 10 12 4 2];
n = sum(obsCounts);
lambdaHat = sum(bins.*obsCounts)/n;
expCounts = n*poisspdf(bins,lambdaHat);
[H6,p6,stat6] = chi2gof(bins,'ctrs',bins,'frequency',obsCounts, ...
                    'expected',expCounts,'nparams',1)
% End of program
% --------------------------------------------------------------------------
```

MATLAB함수 chi2gof.m의 첫 세 가지 사용법들은 관찰값들이 모두 평균과 모수가 알려지지 않은 정규확률분포에서 독립적으로 추출되었는지를 검정하는 것이다. 두 번째와 세 번째 사용법들에서는 기대값을 입력하는 대신에 진짜확률분포함수를 지정하는 옵션 cdf를 사용한다. 세 경우 모두 카이제곱통계량값은 5.9586, 자유도는 $7-1-2=4$ 그리고 p값은 0.2023이다. 따라서, 이 관찰값들이 정규확률분포를 따른다는 귀무가설을 채택한다. 정규확률분포에서 평균과 표준편차를 추정했으므로, 자유도에서 $q=2$를 더 뺐음에 유의하라.

MATLAB함수 chi2gof.m의 네 번째 사용법은 관찰값들이 표준정규확률분포에서 독립적으로 추출되었는지를 검정하는 것이다. 두 번째와 세 번째 사용법들과 마찬가지로 옵션 cdf를 사용했으나 평균과 표준편차를 추정하지는 않는다. 이 명령문을 실행한 결과, 카이제곱통계량값은 5.3981, 자유도는 $7-1=6$ 그리고 p값은 0.4939이다. 따라서, 이 관찰값들이 정규확률분포에서 생성되었다는 귀무가설을 채택한다.

MATLAB함수 chi2gof.m의 다섯 번째 사용법은 관찰값들이 일양확률분포에서 독립적으로 추출되었는지를 검정하는 것이다. 이 경우에는 진짜확률분포함수를 지정하는 옵션 cdf를 사용하는 대신에 기대값들을 포함하는 벡터 expectedCounts를 사용한다. 즉, 소구간들의 경계를 포함하는 벡터 edges를 사용해서 기대값벡터 expectedCounts를 구한다. 이 명령문을 실행한 결과, 카이제곱통계량값은 8, 자유도는 9 그리고 p값은 0.5431이다. 따라서, 이 관찰값들이 일양확률분포에서 생성되었다는 귀무가설을 채택한다.

MATLAB함수 chi2gof.m의 여섯 번째 사용법은 관찰값들이 Poisson확률분포에서 독립적으로 추출되었는지를 검정하는 것이다. 이 경우에는 관찰값들을 직접 사용하는 것이 아니라 관찰값들의 개수들로 구성된 벡터를 사용한다. 따라서, 관찰값들을 포함하는 벡터 대신에 소구간들을 나타내는 벡터 bins를 지정한다. 옵션 ctrs는 소구간들의 중앙값들로 구성된 벡터를 지정하는 것이다. 여기서는 벡터 bins가 사용되었다. 옵션 freqency에는 관찰값들의 개수들로 구성된 벡터 obsCounts를 지정하고, 옵션 expected에는 expCounts를 지정한다. 이 명령문을 실행한 결과, 카이제곱통계량값은 2.5550, 자유도는 3 그리고 p값은 0.4654이다. 따라서, 이 관찰값들이 Poisson확률분포에서 생성되었다는 귀무가설을 채택한다. ■

예제 1.2.4 R함수 chisq.test를 사용해서 적합성을 하는 방법을 살펴보기 위해서, 다음 R 프로그램 ChiSquareTest103R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: ChiSquareTest103R.R
# Goodness of Fit Test
# ------------------------------------------------------------------------
# install.packages("MASS")
library(MASS)
head(survey)
levels(survey$Smoke)
( smoke.freq = table(survey$Smoke) )
smoke.prob = c(.05, .80, .08, .07)
chisq.test(smoke.freq, p=smoke.prob)
# ------------------------------------------------------------------------
```

이 R프로그램에서는 R패키지 MASS에서 제공하는 데이터세트 survey의 변수 Smoke에 관한 적합성검정을 한다. R명령문 'levels(survey$Smoke)'를 사용해서 변수 Smoke의 수준들을 출력한 결과는 다음과 같다.

```
Heavy Never Occas Regul
   11   189    19    17
```

즉, 헤비스모커가 11명, 금연자가 189명, 가끔 피는 사람이 19명 그리고 규칙적으로 피는 사람이 17명이다. 이 카테고리들에 속하는 확률들이 다음과 같다는 귀무가설 H_0를 검정하기로 하자.

$$p_1 = 0.05, \;\; p_2 = 0.80, \;\; p_3 = 0.08, \;\; p_4 = 0.07 \tag{1}$$

R함수 chisq.test를 사용해서 카이제곱검정을 한 결과, 카이제곱검정통계량값은 0.069159이다. 또한, 자유도는 3이고, p값은 0.9953이다. 따라서, 이 귀무가설이 채택된다. ■

1.2.3 Kolmogorov-Smironv검정

서로 독립인 확률변수들 $x_1, x_2, \cdots, x_n$이 누적확률분포함수 $F(x)$를 따른다고 가정하고, 경험확률분포를 다음과 같이 정의하자.

$$F_n(x) \doteq \frac{1}{n} \sum_{j=1}^{n} 1(x_j \le x) \tag{1.2.4}$$

다음 식이 성립함은 자명하다.

$$F_n(x) - F(x) = \frac{1}{n}\sum_{j=1}^{n}[1(x_j \le x) - E(1(x_j \le x))] \tag{1.2.5}$$

식 (1.2.5)와 중심극한정리에 의해서, 다음 식들이 성립함을 알 수 있다.

$$\sqrt{n}[F_n(x) - F(x)] \xrightarrow{d} N(0, F(x)[1 - F(x)]) \tag{1.2.6}$$

대수법칙에서 알 수 있듯이, 고정된 x에 대해서 다음 식이 성립한다.

$$F_n(x) \to P(x_1 \le x) = F(x) \; in \; P \tag{1.2.7}$$

여기서 '$in \quad P$' 는 '확률적으로' 를 의미한다. 식 (1.2.7)에서 알 수 있듯이, 다음 식이 성립한다.

$$\sup_{x\in R}|F_n(x) - F(x)| \to 0 \; in \; P \tag{1.2.8}$$

각 $j\,(= 1, 2, \cdots, n)$에 대해서 다음과 같은 확률변수를 정의하자.

$$u_j \doteq F(x_j) \tag{1.2.9}$$

만약 $F(x)$가 연속이면, 다음 식이 성립한다.

$$P\left(\sup_{x\in R}|F_n - F(x)| \le t\right) = P\left(\sup_{y\in[0.1]}\left|\frac{1}{n}\sum_{j=1}^{n}1(u_j \le y) - y\right| \le t\right) \tag{1.2.10}$$

식 (1.2.10)의 우변은 누적확률분포함수 $F(x)$에 의존하지 않는다. 또한, 다음 식이 성립함이 알려져 있다.

$$\lim_{n\to\infty} P\left(\sqrt{n}\sup_{x\in R}|F_n(x) - F(x)| \le t\right) = H(t) \tag{1.2.11}$$

여기서 Kolmogorov-Smirnov 누적확률분포함수 $H(t)$를 다음과 같이 정의한다.

$$H(t) \doteq 1 - 2\sum_{k=1}^{\infty}[-1]^{k-1}\exp(-2k^2t) \tag{1.2.12}$$

확률분포함수 $H(t)$는 누적확률분포함수 $F(x)$에 의존하지 않는다. 식 1.2.11의 증명은 Doob [15]를 참조하라. 이 증명에서 사용되는 기법을 더블베리어옵션(double barrier option)의

가치를 평가하는데 사용할 수 있다. 식 (1.2.11)을 이용해서, 다음과 같은 가설들을 검정할 수 있다.

$$H_0 : \ P = P_0 \quad vs \quad H_1 : \ P \neq P_0 \tag{1.2.13}$$

이러한 검정법을 Kolmogorov-Smirnov검정이라 한다.

MATLAB함수 kstest.m을 사용하면 Kolmogorov-Smirnov검정을 할 수 있다. 다음 예제들을 살펴보자.

예제 1.2.5 MATLAB함수 kstest.m을 사용해서 Kolmogorov-Smirnov검정을 하기 위해서, 다음 MATLAB프로그램 KStest101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: KStest101.m
%  Kolmogorov-Smirnov test 1
%  Programmed by MATLB
%------------------------------------------------------------------
clear all, close all
rng((1001^2+1)/2,'twister')
ydum = -1.5:0.3:1.5;
yran = ydum.^3;
[H,P_value,KSstat,CV] = kstest(yran)
xdum = -3.5:0.1:3.5;
F = cdfplot(yran);            % Empirical Distribution Function
set(gca,'fontsize',11,'fontweigh','bold')
hold on
G = plot(xdum,normcdf(xdum),'k-');
set(F,'LineWidth',2), set(G,'LineWidth',2)
legend([F G],'EDF','Normal Distribution','location','SE')
saveas(gcf,'KStest101','epsc')
save('KStest101','H','P_value','KSstat','CV')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 수열 $\{-1.5^3, -1.2^3, \cdots, 1.2^3, 1.5^3\}$을 벡터 yran에 저장한다. 이 수열은 결코 정규확률분포에서 추출된 난수들이 아니다. MATLAB함수 kstest의 입력모수는 Kolmogorov-Smirnov검정을 하고자 하는 관찰값들의 벡터 yran이다. 이 명령문을 실행한 결과, 검정통계량값은 0.1418이고 유의수준이 0.05인 임계값(critical value)은 0.3912이며 p값은 0.9579이다. 따라서, 관찰값들이 표준정규확률분포에 생성된 것이라는 귀무가설을 채택한다. 이 결과는 관찰값들의 개수가 적은 경우, Kolmogorov-Smirnov검정이 그리 유용하지 않음을 보여준다. 이 MATLAB프로그램을 실현하면, 경험확률분포와 표준정규확률분포함수를 그린 그림 1.2.3이 그려진다. 그림 1.2.3에서 청색 계단함수는 경험확률분포를 그리고 (흑색) 곡선은 표준정규확률분포함수를 나타낸다. 이 두 함수들의 최대차이가 Kolmogorov-Smirnov검정통계량값이다. ■

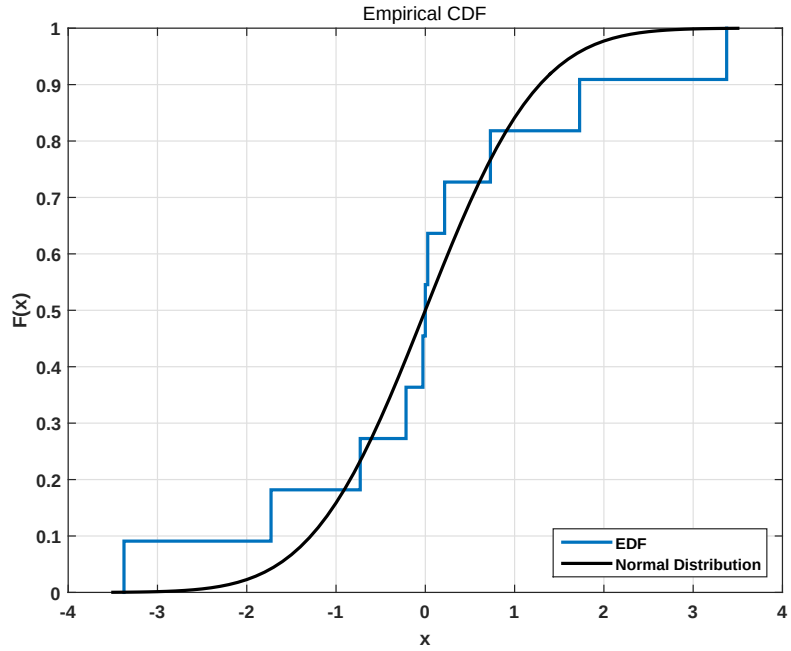


그림 1.2.3. 히스토그램과 카이제곱검정

예제 1.2.6 MATLAB함수 kstest.m의 일반적인 사용법을 알기 위해서, 다음 MATLAB프로그램 KStest102.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: KStest102.m
%  Kolmogorov-Smirnov test 2
%  Programmed by MATLB
%------------------------------------------------------------------
clear all, close all
rng((1001^2+1)/2,'twister')
n = 100
yran = 30 + 2*randn(n,1);              % Normal Random Numbers
meanyran = mean(yran), stdyran = std(yran)
EDF = normcdf(yran,meanyran,stdyran);
[H,P_value,KSstat,CV] = kstest(yran,[yran,EDF],0.025,'unequal')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 정규확률분포 $\mathcal{N}(30, 2^2)$에서 생성된 정규난수들 100개를 벡터 yran에 저장한다. 이 난수들의 표본평균은 29.7460이고 표본표준편차는 1.8894이다. 경험확률분포를 구하기 위해서 MATLAB함수 normcdf를 사용하였고, 이 경험확률분포가 벡터 EDF에 저장된다. MATLAB함수 kstest의 첫 번째 입력모수 yran은 Kolmogorov-Smirnov검정을 하고자 하는 관찰값들의 벡터이고, 두 번째 입력모수는 경험확률분포이다. 이 경험확률분포는 반드시 관찰벡터를 함께 입력해야 한다. 세 번째 입력모수는 유의수준이다. 유의수준의 디폴트는 0.05이고 이 MATLAB프로그램에서는 0.025를 사용한다. 마지막

입력모수는 양측검정이냐 단측검정이냐를 지정하는 것이다. 만약 'unequal' 을 사용하면, 대립가설은 모집단의 누적확률분포함수가 정규확률분포함수과 다르다는 것이다. 이것이 디폴트이다. 만약 'smaller' 를 사용하면, 대립가설은 모집단의 누적확률분포함수가 정규확률분포함수보다 작다는 것이다. 만약 'larger' 를 사용하면, 대립가설은 모집단의 누적확률분포함수가 정규확률분포함수보다 크다는 것이다. 이 명령문을 실행한 결과, 검정통계량값은 0.0659이고 유의수준이 0.025인 임계값은 0.1462이며 p값은 0.7519이다. 따라서, 관찰값들이 표준정규확률분포에 생성된 것이라는 귀무가설을 채택한다. ■

Kolmogorov-Smirnov검정법은 주어진 확률분포 P_0에 대한 단순가설을 검정하는 것이다. 그러나, 예제 1.2.6의 경우에는 확률분포 P_0가 미리 주어진 것이 아니라 추정한 것이다. 즉, 귀무가설 하에서 확률분포가 $\mathcal{N}(\mu, \sigma^2)$이 아닌 추정확률분포 $\mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$이다. 이러한 경우에는 Lilliefors [27], [28]가 Kolmogorov-Smirnov검정통계량을 수정해서 제시한 Lilliefors검정통계량이 좀 더 정확하다.

예제 1.2.7 MATLAB함수 lillietest.m의 사용법을 알기 위해서, 다음 MATLAB프로그램 LillieforsTest101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: LillieforsTest101.m
%  Lilliefors Test 1
%  Programmed by MATLB
%-------------------------------------------------------------------
clear all, close all
rng((1001^2+1)/2,'twister')
n = 100
yran = 30 + 2*randn(n,1);              % Normal Random Numbers
meanyran = mean(yran), stdyran = std(yran)
[H,P_value,Lstat,CV] = lillietest(yran,0.025,'norm')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 정규확률분포 $\mathcal{N}(30, 2^2)$에서 생성된 정규난수들 100개를 벡터 yran에 저장한다. 이 난수들의 표본평균은 30.1143이고 표본표준편차는 1.7380이다. MATLAB함수 lillietest의 첫 번째 입력모수 yran은 Lilliefors검정을 하고자 하는 관찰값들의 벡터이고, 두 번째 입력모수는 유의수준이다. 유의수준의 디폴트는 0.05이고 이 MATLAB프로그램에서는 0.025를 사용한다. 마지막 입력모수는 귀무가설 하에서 확률분포의 종류로서, 정규확률분포인 'norm', 지수분포인 'exp' 그리고 극단확률분포(extreme value distibution)인 'ev' 중에서 선택할 수 있다. 그러나, 대수정규확률변수의 로그가 정규확률변수라는 성질 또는 Weibull확률변수의 로그가 극단확률변수라는 성질을 사용해서

관찰값들이 대수정규확률분포 또는 Weibull확률분포에서 발생되었다는 귀무가설들도 검정할 수 있다. 이 명령문을 실행한 결과, 검정통계량값은 0.0420 이고 유의수준이 0.025 인 임계값은 0.0956 이며 p 값은 수표에서 제공한 값보다 크므로 0.5000 을 출력한다. 따라서, 관찰값들이 표준정규확률분포에 생성된 것이라는 귀무가설을 채택한다. ■

예제 1.2.8 R함수 ks.test를 사용해서 적합성을 하는 방법을 살펴보기 위해서, 다음 R프로그램 KStest104R.R을 실행해 보자.

```
# ------------------------------------------------------------------------------
# Filename: KStest104R.R
# Kolmogorov-Smirnov Test
# Programmed by CBS
# ------------------------------------------------------------------------------
% Making Dataset
yran1 <- rnorm(60,mean=170,sd=10)
yran2 <- rnorm(40,mean=155,sd=8)
yran <- matrix(100,1)
yran[1:60] <- yran1
yran[61:100] <- yran2
ysex <- matrix(100,1)
yran[61:100] <- yran2
ysex[1:60] <-"M"
ysex[61:100] <- "F"

# Kolmogorov-Smirnov Test
ks.test(yran1,yran2)

# Plotting
# install.packages("ggplot2")
library(ggplot2)
setEPS()
plot.new()
postscript('KStest104R.eps')  # Start to save figure
KSdata1 <- data.frame(yran,ysex)
ggplot(KSdata1, aes(yran, fill=ysex)) +
       geom_histogram(aes(y = ..density..),position='identity')
dev.off()             # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램에서는 $\mathcal{N}\left(170, 10^2\right)$ 에서 생성된 정규난수들 60 개를 벡터 yran1에 저장하고, $\mathcal{N}\left(155, 8^2\right)$ 에서 생성된 정규난수들 40 개를 벡터 yran2에 저장한다. 여기서 yran1은 울릉대학교 남자신입생의 키를 나타내고, yran2는 여자신입생의 키를 나타낸다고 하자. 성별은 확률변수 ysex에 M과 F로 표기한다.

R함수 ks.test를 사용해서 Kolmogorov-Smirnov검정을 한 결과, ks검정통계량값은 0.59167이다. 또한, p값은 $2.86 \cdot 10^8$ 이다. 따라서, 확률변수들 yran1과 yran2가 동일한 확률분포를 따른다는 귀무가설을 기각한다. 이 R프로그램을 실현하면, 성별로 히스토그램을

그린 그림 1.2.4이 그려진다. 그림 1.2.4에서 두 집단이 동일한 확률분포를 따른다고 가정하는 것이 무리라는 것을 알 수 있다. ■

그림 1.2.4. KS검정과 히스토그램

제1.3절 역함수법

일양확률분포가 아닌 확률분포를 따르는 난수를 생성하는 일반적인 방법으로는 역함수법(inverse transformation method)과 채택기각법(accept-reject method)이 있다. 여기서는 역함수법을 다루고, 다음 절에서 채택기각법을 살펴보자.

연속형 확률변수 x의 누적확률분포함수 $F(x)$에 대해서 확률변수 $u \doteq F(x)$는 다음 식들을 만족한다.

$$Pr(u \leq a) = Pr(F(x) \leq a) = Pr(x \leq F^{-1}(a)) = F(F^{-1}(a)) = a \tag{1.3.1}$$

즉, 확률변수 u는 일양확률분포를 따른다. 변수변환 $u = F(x)$를 확률적분변환(probability integral transformation)이라 부른다. 식 1.3.1에서 알 수 있듯이, 강단조증가하는(strictly

increasing) 누적확률분포함수 $F(\cdot)$와 지지대가 $(0,1)$인 일양확률변수 u에 대해서 $x \doteq F^{-1}(u)$는 누적확률분포함수 $F(x)$를 따르는 연속형 확률변수이다. 이 성질을 이용해서 주어진 확률분포에 따르는 난수를 생성하는 것을 역함수법 (inverse transformation method)이라 한다. 만약 누적확률분포함수 $F(\cdot)$가 강단조증가하지 않거나 연속이 아니면, 역함수 대신에 다음과 같은 일반화역함수를 사용한다.

$$F^{\leftarrow}(u) \doteq \min\{x \,|\, F(x) \geq u\} \tag{1.3.2}$$

예제 1.3.1 다음과 같은 확률밀도함수를 살펴보자.

$$f(x) = 2x1_{[1,0]}(x) \tag{1}$$

이에 해당하는 누적확률분포함수는 다음과 같다.

$$F(x) = \int_{-\infty}^{x} f(x)dx = x^2 1_{[0,1)}(x) + 1_{[1,-\infty)}(x) \tag{2}$$

확률분포함수 $F(x)$의 역함수는 다음과 같다.

$$F^{-1}(u) = \sqrt{u} \tag{3}$$

따라서, $\{u_1, u_2, \cdots, u_n\}$을 지지대가 $(0,1)$인 일양난수들이라 하면, $\left\{x_j \doteq \sqrt{u_j} \mid j = 1, 2, \cdots, n\right\}$는 삼각형 확률밀도함수 (1)을 따르는 난수들이다. ■

예제 1.3.2 평균이 λ인 지수확률분포의 누적확률분포함수는 다음과 같다.

$$F(x) = 1 - \exp\left(-\frac{x}{\lambda}\right), \quad (x \geq 0) \tag{1}$$

함수 $F(x)$의 역함수는 다음과 같다.

$$F^{-1}(u) = -\lambda \ln(1-u) \tag{2}$$

확률변수 u가 지지대가 $(0,1)$인 일양확률분포를 따르면, $1-u$도 같은 일양확률분포에 따른다. 따라서, $\{u_1, u_2, \cdots u_n\}$을 지지대 $(0,1)$상에서 일양난수들이라 하면, $\{x_j = -\lambda \ln u_j\}$는

평균값이 λ인 지수확률분포를 따르는 난수들이다.

역함수법을 적용해서 평균값이 $\lambda = 0.5$인 지수확률분포로부터 난수들을 생성하기 위해서, 다음 MATLAB프로그램 InverseMethodExponential101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: InverseMethodExponential101.m
%  Exponential Random Numbers by Inverse Method
%  Programmed by CBS
%-------------------------------------------------------------------------
clear all, close all
rng((1001^2+1)/2,'twister')
N = 10000;
lambda = 0.5                          % Exponential w/ mean mu = 0.5
yran = -0.5*log(rand(1,N));
% Goodness-of-Fit Test
[H,p_value] = chi2gof(yran,'cdf',{@expcdf,lambda})
% Plotting Histogram and PDF
[ Nbar,xcenter ] = hist(yran,21);
Nheight = Nbar/( xcenter(2)-xcenter(1) )/N;
bar(xcenter,Nheight,1,'w')
set(gca,'fontsize',11,'fontweigh','bold')
hold on
xx = linspace(0,6,1001);
pp = 2*exp(-2*xx);                    % True PDF
plot(xx,pp,'r','linewidth',2)
legend('Histogram','True PDF',1)
xlabel('\bf x'), ylabel('\bf Relative Frequency')
axis([-0.2 6.2 0 2.5])
hold off
saveas(gcf,'InverseMethodExponential101','epsc')
save('InverseMethodExponential101','yran','H','p_value')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 평균이 $\lambda = 0.5$인 지수확률분포로부터 난수들 10000개가 생성된다. 카이제곱검정법을 사용해서 적합성검정을 한 결과 p값이 0.7547이다. 따라서, 이 난수들이 이 지수확률분포로부터 생성되었다는 귀무가설이 채택된다. 이 MATLAB프로그램을 실행하면, 그림 1.3.1이 출력된다. 그림 1.3.1에서 막대그래프는 이 난수들의 히스토그램이고, 적색 점들은 진짜확률밀도함수값들이다. 그림 1.3.1에서 이 난수들이 지수확률밀도함수 $f(x) = 2\exp(-2x)1_{(x\geq 0)}$에서 발생되었다는 것을 확인할 수 있다. ■

예제 1.3.3 다음과 같이 모수가 θ인 로지스틱확률분포함수를 살펴보자.

$$F(x) = \frac{1}{1+\exp(-\theta x)} \tag{1}$$

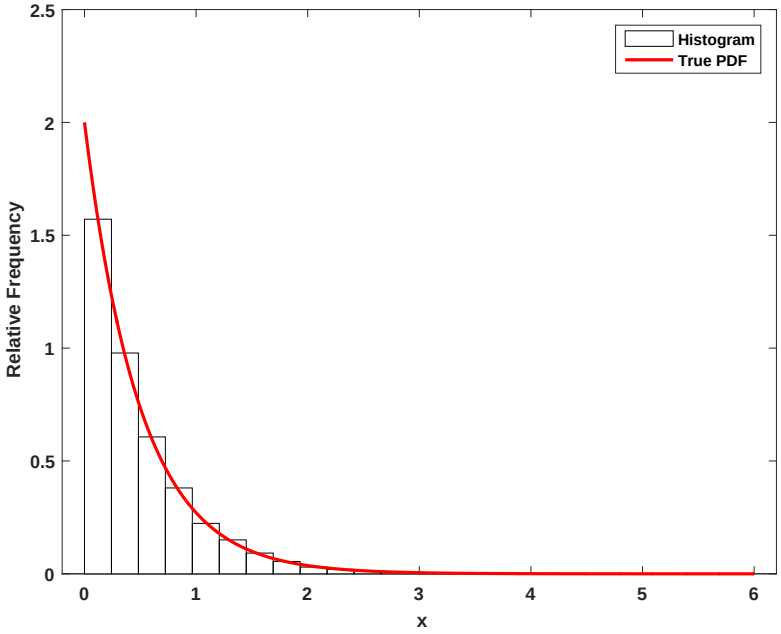


그림 1.3.1. 역함수법과 지수확률분포

함수 $F(x)$의 역함수는 다음과 같다.

$$F^{-1}(u) = -\frac{1}{\theta}\ln\left(\frac{1-u}{u}\right) \tag{2}$$

따라서, $\{u_1, u_2, \cdots, u_n\}$을 지지대를 $(0,1)$으로 하는 일양난수들이라 하면,

$\left\{x_j \doteq -\frac{1}{\theta}\ln\left(\frac{1-u}{u}\right) \mid j = 1, 2, \cdots, n\right\}$은 모수가 θ인 로지스틱확률분포를 따르는 난수들이다.

역함수법을 적용해서 모수가 $\theta = 2$인 로지스틱확률분포로부터 난수들을 생성하기 위해서, 다음 MATLAB프로그램 InverseMethodLogistic101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: InverseMethodLogistic101.m
%  Logistic Random Numbers by Inverse Method
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
rng((1001^2+1)/2,'twister')
N = 100
uu = rand(1,N);
yran = -0.5*log((1-uu)./uu); yran = yran';
% Kolmogorov-Smirnov Test
CDF = normcdf(yran,mean(yran),std(yran,1));
[H,P_value,KSstat,CV] = kstest(yran,[yran,CDF],0.05)
% Plotting Empirical Distribution and CDF
F = cdfplot(yran);              % Empirical Distribution Plot
set(F,'color','k','LineWidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
hold on
xx = -6:0.1:6;
cc = 1./(1+exp(-2*xx));
```

```
G = plot(xx,cc,'r--','linewidth',2);            % True CDF
legend('Empirical Dist','True CDF','location','SE')
xlabel('\bf x'), ylabel('\bf Relative Cumulative Frequency')
axis([-6 6  0 1.05 ])
hold off
saveas(gcf,'InverseMethodLogistic101','epsc')
save('InverseMethodLogistic101','yran','H','P_value')
% End of program
% --------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 평모수가 $\theta = 2$인 로지스틱확률분포로부터 난수들 100개가 생성된다. Kolmogorov-Smirnov검정법을 사용해서 적합성검정을 한 결과, Kolmogorov-Smirnov검정통계량값은 0.0368이고, 임계값(cutting-off value; CV)은 0.1340이며, p값이 0.9986이다. 따라서, 이 난수들이 이 로지스틱확률분포로부터 생성되었다는 귀무가설이 채택된다. 이 MATLAB프로그램을 실행하면, 그림 1.3.2가 출력된다. 그림 1.3.2에서 (흑색) 실선은 이 난수들의 경험확률분포(empirical distribution: EDF)이고, 적색 긴점선은 진짜확률분포함수이다. 그림 1.3.2에서 이 난수들이 로지스틱확률밀도함수에서 발생되었다는 것을 확인할 수 있다. ■

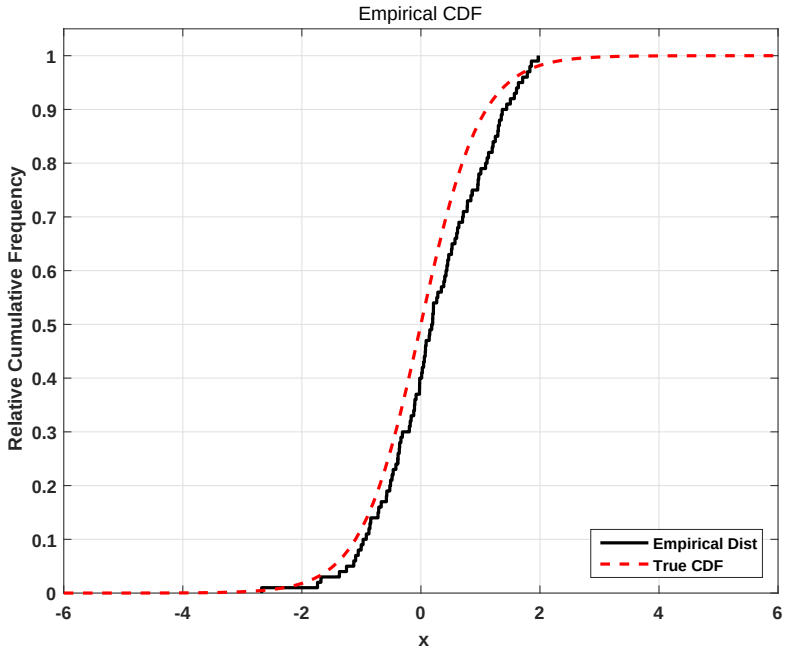


그림 1.3.2. 역함수법과 로지스틱확률분포

예제 1.3.4 역함수법을 적용해서 모수가 정규확률분포로부터 난수들을 생성하기 위해서, 다음 R프로그램 InverseMethodNormal101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
```

```
# Filename: InverseMethodNormal101R.R
# Goodness of Fit Test
# Progtrmmed by CBS
# ------------------------------------------------------------------------
# Generating Normal Random Numbers
InverseSampler <- function(n){
                     NormalRV <- NULL
                     for (ii in 1:n){
                          Xunif <- runif(1,min=0,max=1)
                          Ynormal <- qnorm(Xunif,mean=0,sd=1)
                          NormalRV <- c(NormalRV,Ynormal)
                     }
                     return(NormalRV)
                   }
nSim <- 500
set.seed(11)
NormRV <- InverseSampler(5000)

# Plotting
library(ggplot2)
setEPS()
plot.new()
postscript('InverseMethodNormal101R.eps')  # Start to save figure
IMdata1 <- data.frame(NormRV)
ggplot(IMdata1, aes(x=NormRV)) +
      geom_histogram(aes(y=..density..),binwidth=.1,
                     colour="black", fill="yellow") +
      stat_function(fun=dnorm,args=list(mean=0,sd =1),lwd=1.1,color='red') +
      xlab(expression(x))
dev.off()             # End to save figure
# ------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 표준정확률분포로부터 난수들 5000개가 생성되고 또한 그림 1.3.3가 출력된다. 그림 1.3.3에는 생성된 난수들의 히스토그램과 더불어 표준정규확률밀도함수가 그려져 있다. 그림 1.3.3에서 알 수 있듯이, 생성된 난수들의 히스토그램은 표준정규확률밀도함수와 아주 가깝다. ■

예제 1.3.5 공정한 주사위를 던지는 실험을 시뮬레이션하기로 하자. 즉, 지지대가 $\{1,2,3,4,5,6\}$인 일양확률분포를 따르는 난수들을 발생시키자. 이 이산형 일양확률분포의 누적확률분포함수는 다음과 같다.

$$F(x) = \sum_{j=1}^{\min\{6,\lfloor x \rfloor\}} \frac{1}{6}, \quad (x \geq 0) \tag{1}$$

여기서 $\lfloor w \rfloor$는 w를 넘지 않는 최대 정수이다. 함수 $F(x)$의 일반화역함수는 다음과 같다.

$$F^{\leftarrow}(u) = \lceil 6u \rceil \quad (0 < u \leq 1) \tag{2}$$

여기서 $\lceil w \rceil$는 w와 같거나 초과하는 최소 정수이다.

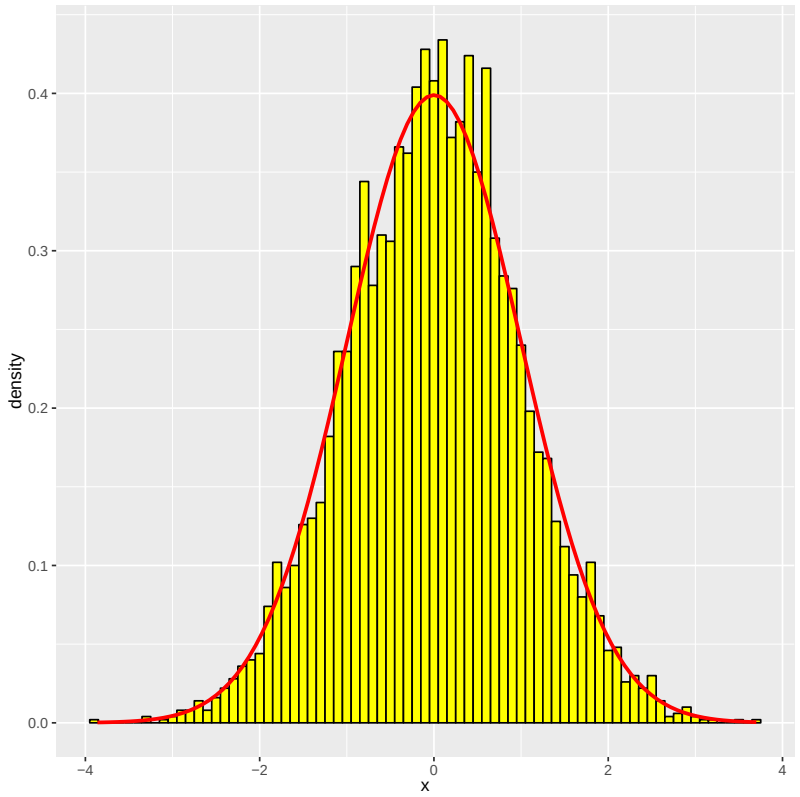


그림 1.3.3. 역함수법과 정규확률분포

역함수법을 적용해서 이 이산형 일양확률분포로부터 난수들을 생성하기 위해서, 다음 MATLAB프로그램 InverseMethodDiscreteUniform101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: InverseMethodDiscreteUniform101.m
%  Discrete Uniform Random Numbers using Inverse Method
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all
rng((1001^2+1)/2,'twister')
N = 1000
uu = rand(1,N);
yran = ceil(6*uu);
% Chi-squared Test
edges = 0.5:1:6.5;
expectedCounts = N/6*ones(1,6);
[H,P_value,stat] = chi2gof(yran,'edges',edges,'expected',expectedCounts)
% Plotting Histogram and True PDF
xx = 1:6;
yFreq = hist(yran,xx)/N;
bar(xx,yFreq,0.2,'w')
hold on
set(gca,'fontsize',11,'fontweigh','bold')
plot(1:6,1/6*ones(1,6),'r*','linewidth',2)
legend('Histogram','True PMF','location','NE')
xlabel('\bf x'), ylabel('\bf Relative Frequency')
axis([0.6 6.4 0 0.3])
hold off
```

```
saveas(gcf,'InverseMethodDiscreteUniform101','epsc')
save('InverseMethodDiscreteUniform101','yran','H','P_value')
% End of program
% --------------------------------------------------------------------------
```

이 MATLAB 프로그램을 실행하면, 지지대가 $\{1, 2, 3, 4, 5, 6\}$ 인 일양확률분포를 따르는 난수들을 10000 개가 생성된다. MATLAB 함수 chi2gof.m 을 사용해서 적합성검정을 한 결과, 카이제곱통계량은 6.1760 이고 자유도는 5 이며 p 값이 0.2895 로 이 난수들이 이 이산형 일양확률분포로부터 생성되었다는 귀무가설이 채택된다. MATLAB 함수 chi2gof.m 의 옵션 edges는 상자들(bins), 즉 소구간들의 경계를 나타냄에 유의하라. 이 MATLAB 프로그램을 실행하면, 그림 1.3.4이 출력된다. 그림 1.3.4에서 막대그래프는 이 난수들의 히스토그램이고, 적색 실선은 진짜확률밀도함수이다. 그림 1.3.4에서 이 난수들이 이산형 일양확률분포로부터 발생되었다는 것을 확인할 수 있다. ■

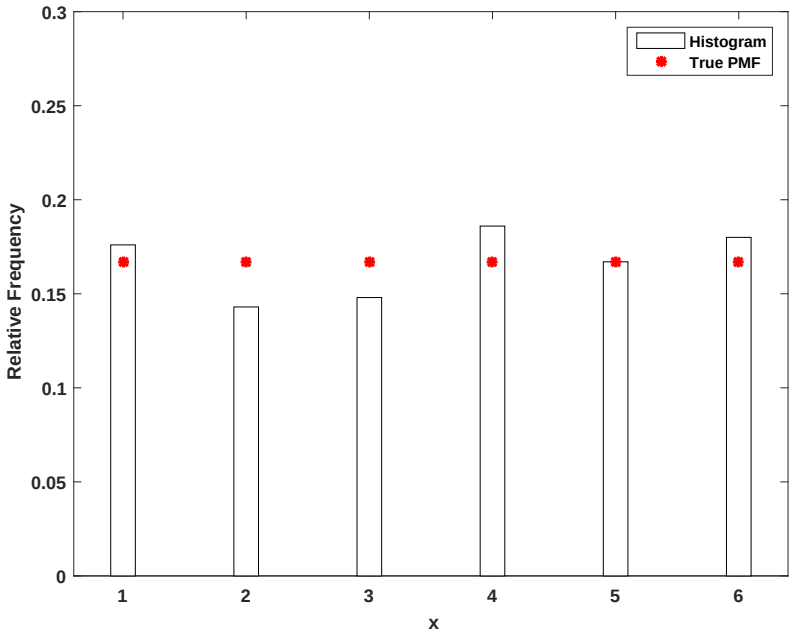


그림 1.3.4. 역함수법과 이산형 일양확률분포

역함수법은 원리는 단순하지만, 역함수 $F^{-1}(x)$ 를 간단한 함수로 표현할 수 없거나 계산에 시간이 많이 걸리는 경우에는 좋은 방법이라고 할 수는 없다.

제1.4절 채택기각법

다음과 같은 기대값을 계산하기로 하자.

$$E_f(g(x)) \doteq \int_{-\infty}^{\infty} g(x)f(x)dx \tag{1.4.1}$$

함수 $g(x)$와 지지대가 같은 확률밀도함수 $h(x)$를 이용해서, 이 정적분을 다음과 같이 쓸 수 있다.

$$E_f(g(x)) = \int_{\infty}^{\infty} \frac{g(x)f(x)}{h(x)}h(x)dx = E_h\left(\frac{g(x)f(x)}{h(x)}\right) \tag{1.4.2}$$

만약 함수 $g(x)f(x)/h(x)$가 상수에 가깝도록 $h(x)$를 선택할 수 있고, 또한 만약 확률밀도함수 $h(x)$로부터 난수를 생성하는 것이 쉽다면, 식 (1.4.2)를 사용해서 정적분 $E_f(g(x))$를 추정하는 것이 좋다. 확률밀도함수 $h(x)$로부터 생성된 난수들을 $\{x_j \mid j = 1, 2, \cdots, n\}$이라 하면, 다음 근사식이 성립한다.

$$E_f(g(x)) \approx \frac{1}{n}\sum_{j=1}^{n} \frac{g(x_j)f(x_j)}{h(x_j)} \tag{1.4.3}$$

난수를 생성하려고 하는 확률분포의 확률밀도함수와 지지대를 각각 $f(x)$와 S라고 하자. 지지대가 S이고 난수의 생성이 용이한 어떤 확률밀도함수를 $g(x)$라 하자. 채택기각법은 이 확률밀도함수 $g(x)$에서 발생한 난수를 이용해서 확률밀도함수 $f(x)$로부터 난수를 생성하는 방법이다. 채택기각법을 기각법이라고도 부른다. 각 $x(\in S)$에 대해서 다음 식을 만족하는 상수 c가 존재한다고 가정하자.

$$\frac{f(x)}{g(x)} < c \tag{1.4.4}$$

일양확률변수 u와 확률밀도함수가 $g(\cdot)$인 확률변수 y에 대해서 다음 식들이 성립한다.

$$\begin{aligned}
&Pr\left(u \le \frac{f(y)}{cg(y)}\right) \\
&= \int_0^{\infty}\int_0^{f(y)/[cg(y)]} 1 \cdot g(y)dudy \\
&= \int_{-\infty}^{\infty} Pr\left(u \le \frac{f(y)}{cg(y)}\right) g(y)dy \\
&= \int_{-\infty}^{\infty} \frac{f(y)}{cg(y)}g(y)dy \\
&= \frac{1}{c}\int_{-\infty}^{\infty} f(y)dy = \frac{1}{c}
\end{aligned} \tag{1.4.5}$$

다음 식들이 성립한다.

$$f(x)dx = \frac{g(x)dx \cdot \frac{f(x)}{cg(x)}}{\frac{1}{c}} = \frac{Pr\left(x \le y < x+dx,\ u \le \frac{f(y)}{cg(y)}\right)}{Pr\left(u \le \frac{f(y)}{cg(y)}\right)} \tag{1.4.6}$$

여기서 y는 확률밀도함수가 $g(\cdot)$인 확률변수이고, 두 번째 등호는 식 (1.4.5)에 의해서 성립한다. 식 (1.4.6)에서 알 수 있듯이, 다음 식이 성립한다.

$$f(x)dx = Pr\left(x \le y < x+dx \,\middle|\, u \le \frac{f(y)}{cg(y)}\right) \tag{1.4.7}$$

식 (1.4.5)에서 알 수 있듯이, 채택기각법은 요점추출법 (importance sampling method)의 특수한 경우이다.

식 (1.4.7)에서 알 수 있듯이, 다음 알고리즘을 사용해서 확률밀도함수 $f(x)$를 따르는 난수들을 생성할 수 있다.

알고리즘 1.4.1: 채택기각법

(1단계) 확률밀도함수 $g(x)$에 따르는 난수 y를 생성한다.

(2단계) 일양난수 u를 생성한다.

(3단계) 만약 식 $u < f(y)/[cg(y)]$이 성립하면, y를 확률밀도함수 $f(x)$를 따르는 x로 간주한다. 그렇지 않으면, 제1단계로 돌아간다. 즉, 확률 $f(y)/[cg(y)]$로서 후보값 y를 x로 선택한다.

예제 1.4.1 다음과 같은 베타확률밀도함수를 살펴보자.

$$f(x) = \frac{1}{B(a,b)} x^{a-1}[1-x]^{b-1}, \qquad (0 \le x \le 1) \tag{1}$$

이 베타확률분포는 점 $x = [a-1]/[a+b-2]$에서 다음과 같은 최대값을 갖는다.

$$c = \frac{1}{B(a,b)} \frac{[a-1]^{a-1}[b-1]^{b-1}}{[a+b-2]^{a+b-2}} \tag{2}$$

베타확률분포의 지지대가 지지대 $(0,1)$이므로, 난수를 생성하는 확률밀도함수 $g(x)$를 지지대가 $(0,1)$인 일양확률분포를 따른다고 하자. 이 경우에, 식 (2)의 c를 알고리즘 1.4.1의 c로

사용한다.

채택기각법을 이용해서 베타확률분포로부터 난수들을 생성하기 위해서, 다음 MATLAB 프로그램 AcceptRejectBeta101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: AcceptRejectBeta101.m
%  Generating Beta Random Numbers by Acceptance-Rejection Method
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
rng((1001^2+1)/2,'twister')
N = 400
% set up storage arrays
xacc = zeros(1,N);        % random number
yacc = zeros(1,N);        % corresponding y value
xrej = zeros(1,N);        % rejected random number
yrej = zeros(1,N);        % corresponding y value
iiacc = 1;
iirej = 1;
a = 3, b = 2
BB = beta(a,b)
ff = inline('1/beta(3,2)*x^(3-1)*(1-x)^(2-1)','x')
gg = inline('x+1-x')
c = 1/BB*(a-1)^(a-1)*(b-1)^(b-1)/(a+b-2)^(a+b-2)
% constant
while iiacc <= N
    xdum = rand(1);
    u = rand(1);
    if u <= ff(xdum)/c/gg(xdum)
        xacc(iiacc) = xdum;
        yacc(iiacc) = u*gg(xdum)*c;
        iiacc = iiacc+1;
    else
        xrej(iirej) = xdum;
        yrej(iirej) = u*gg(xdum)*c;
        iirej = iirej+1;
    end
end
% 2D Plot of Accept and Reject Regions
xdum2 = linspace(0,1,1001);
ydum2 = 1/BB*xdum2.^(a-1).*(1-xdum2).^(b-1);
ydum3 = xdum2+1-xdum2;
plot(xacc,yacc,'mo',xrej,yrej,'bd',xdum2,ydum2,'r',xdum2, ...
     ydum3,'k--','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
legend('\bf Accept','\bf Reject','\bf f(x)','\bf g(x)', ...
       'location','NW')
saveas(gcf,'AcceptRejectBeta101a','epsc')
% Goodness-of-Fit test
RN = xacc;                    % RN: Generated random numbers
RNmean = mean(RN), RNvar = var(RN)
dumRN = RNmean*(1-RNmean)/RNvar - 1
ahat = RNmean*dumRN
bhat = (1-RNmean)*dumRN
[H,P_value,stat] = chi2gof(RN,'cdf', ...
                           @(z)betacdf(z,ahat,bhat),'npara',2)
% Plotting Histogram and True PDF
figure
[ Nbar,xcenter ] = hist(RN,21);
```

```
Nheight = Nbar/( xcenter(2)-xcenter(1) )/N;
xx = linspace(0,1,1001);
pp = 1/BB*xx.^(3-1).*(1-xx).^(2-1);   % True PDF
bar(xcenter,Nheight,1,'w')
set(gca,'fontsize',11,'fontweigh','bold')
hold on
plot(xx,pp,'r','linewidth',2)
xlabel('x'), ylabel('Relative Frequency')
hold off
saveas(gcf,'AcceptRejectBeta101b','epsc')
save('AcceptRejectBeta101','xacc','yacc')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 모수벡터가 $(a,b)=(3,2)$ 인 베타확률분포에서 400 개 난수들이 생성되고, 발생된 난수들을 2차원으로 그린 그래프가 그림 1.4.1이다. 그림 1.4.1에서 (분홍색) 원은 채택되는 난수를 나타내고, 청색 다이아몬드는 기각되는 난수를 나타낸다. 또한, 적색 실선은 목적확률밀도함수 $f(x)$ 이고, (흑색) 긴점선은 난수를 발생시키기가 쉬운 확률밀도함수 $g(x)$ 이다. 상수 c는 1.7778 이므로, 확률밀도함수 $g(x)$ 에서 발생한 난수들 중에서 56.25 퍼센트를 채택되고 나머지를 폐기된다.

이 베타난수들로부터 추정된 모수벡터의 추정벡터는 $(\hat{a},\hat{b}) = (3.0329, 1.9908)$ 이다. MATLAB함수 chi2gof.m을 사용해서 적합성검정을 한 결과, 카이제곱통계량은 5.8196 이고 자유도는 7 이며 p값이 0.5610 이다. 따라서 이 난수들이 이 베타확률분포로부터 생성되었다는 귀무가설이 채택된다. 이 MATLAB프로그램을 실행하면, 그림 1.4.2도 출력된다. 그림 1.4.2에서 막대그래프는 이 난수들의 히스토그램이고, 적색 실선은 진짜확률밀도함수이다. 그림 1.4.2에서 이 난수들이 베타확률분포로부터 발생되었다는 것을 확인할 수 있다. ■

채택기각법을 좀 더 자세히 살펴보기 위해서, 다음과 같이 함수의 몸통(body)을 정의하자.

정의 1.4.1

비음이고 적분가능한 함수 f 의 몸통은 다음과 같다.

$$B_f \doteq \{[x,z] \mid 0 \le z \le f(x)\}$$

양수 c에 대해서 확률벡터 $[x,z]$가 함수 cf 의 몸통 B_{cf} 에서 일양확률분포를 따르면, 다음 식들이 성립한다.

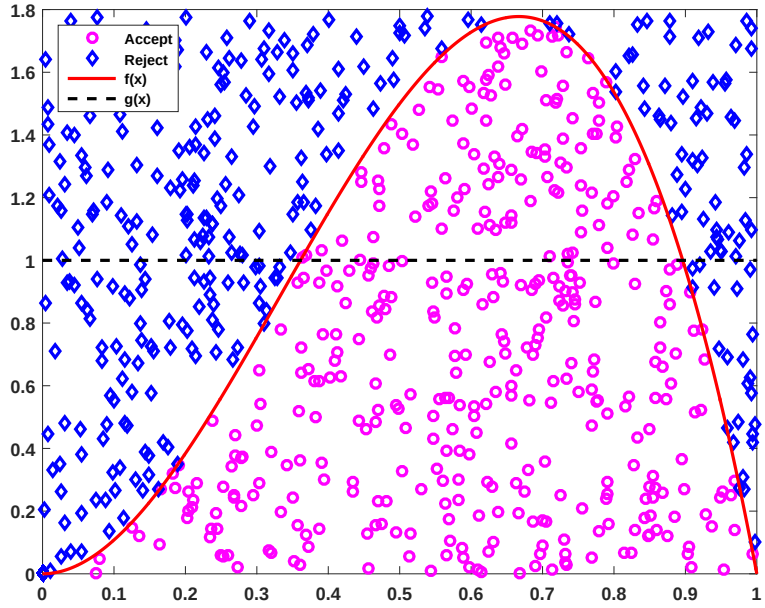


그림 1.4.1. 채택기각법과 베타확률분포

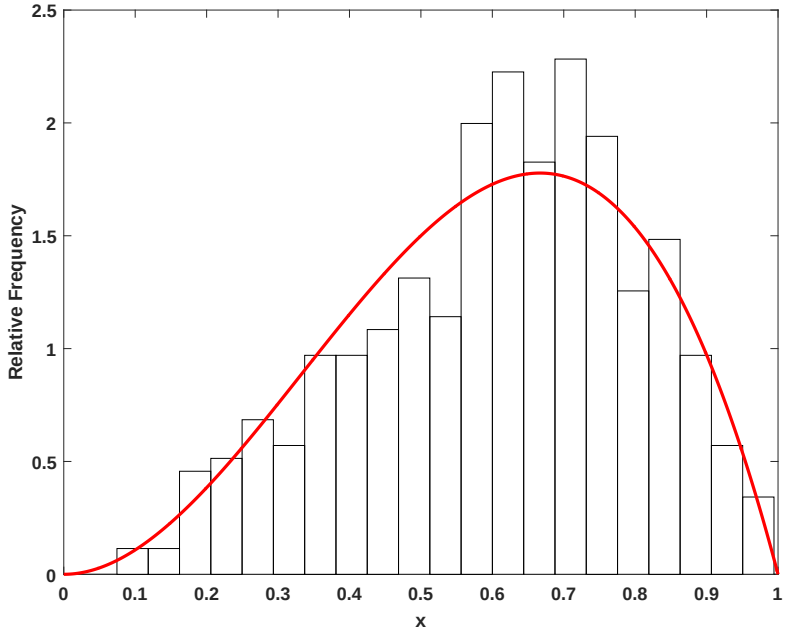


그림 1.4.2. 채택기각법에 의해서 발생한 베타난수들의 히스토그램

$$\frac{\int_x^{x+dx}\int_0^{cf(v)} dzdv}{\int_S \int_0^{cf(v)} dzdv} = \frac{1}{\int_S cf(v)dv}\int_x^{x+dx} cf(v)dv = \int_x^{x+dx} f(v)dv \tag{1.4.8}$$

여기서 S는 함수 f의 지지대이다. 식 (1.4.8)에서 확인할 수 있듯이, 확률변수 x는 확률밀도함수 f를 갖는다. 역으로 다음 정리가 성립한다. 이 정리의 증명은 Devroye [14, p.40]를 참조하라.

정리 1.4.1

확률변수 y는 확률밀도함수 g를 갖고, 지지대가 $[0,1]$인 일양확률변수 u는 확률변수 y와 독립이고, c는 양수라고 하자. 이러한 조건 하에서 확률벡터 $[y, u \cdot c \cdot g(y)]$는 함수 cg의 몸통 B_{cg}에서 일양확률분포를 따른다.

우리의 목표는 주어진 확률밀도함수 f의 몸통 B_f에서 일양난수를 선택하는 것이다. 확률변수 y는 확률밀도함수 g를 갖고, 지지대가 $[0,1]$인 일양확률변수 u는 확률변수 y와 독립이고, 또한 c는 양수라고 하자. 정리 1.4.1에서 알 수 있듯이, 이러한 조건 하에서 확률벡터 $[y, u \cdot c \cdot g(y)]$는 함수 cg의 몸통 B_{cg}에서 일양확률분포를 따른다. 만약 $u \cdot c \cdot g(y) \leq f(y)$이면, $[y, u \cdot c \cdot g(y)]$는 함수 f의 몸통 B_f에 속한다. 확률벡터 $[y, u \cdot c \cdot g(y)]$가 함수 f의 몸통 $B_f\,(\subset B_{cg})$에서 일양확률분포를 따르므로, 식 (1.4.8)에서 알 수 있듯이 확률변수 y는 확률밀도함수 f를 갖는다. 이것이 채택기각법의 핵심이다.

예제 1.4.2 지지대(support)가 Ω인 확률변수가 특정한 구간 $A(\subset \Omega)$에서만 관찰될 때, 그 확률분포를 구간 A밖에서 절단확률분포라 한다. 원래 확률분포의 확률밀도함수를 $f(x)$라고 하면, 절단확률분포의 확률밀도함수 $h(x)$는 다음과 같다.

$$h(x) = \frac{1}{Pr(A)} f(x) 1_A(x) \tag{1}$$

지금부터는 표준정규확률분포에 따르는 확률변수의 지지대를 구간 (a,b)로 제한한 확률분포에서 난수들을 생성하기로 하자. 간단한 방법으로는, 우선 표준정규난수들을 생성하고, 그 중에서 구간 (a,b)안에 들어가는 것만을 취하면 된다. 그러나, 여기서는 채택기각법을 사용하기로 하자. 이 절단정규확률분포의 확률밀도함수는 다음과 같다.

$$f(x) = \frac{1}{N(b) - N(a)} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x^2\right) 1_{(a,b)}(x) \tag{2}$$

단, 식 $ab < 0$가 성립한다고 가정하자.

구간 (a,b)상에서 일양확률밀도함수를 $g(x)$라 하면, 다음 식이 성립한다.

$$g(x) = \frac{1}{b-a} 1_{(a,b)}(x) \tag{3}$$

확률밀도함수 $f(x)$의 최대값은 다음과 같다.

$$f(0) = \frac{1}{N(b) - N(a)} \frac{1}{\sqrt{2\pi}} \tag{4}$$

다음과 같이 상수 c를 정의하자.

$$c = \frac{f(0)}{g(0)} = \frac{1}{N(b) - N(a)} \frac{1}{\sqrt{2\pi}} [b - a] \tag{5}$$

따라서, 확률밀도함수 $g(x)$를 따르는 난수 y를 채택할 확률은 다음과 같다.

$$\frac{f(y)}{c \cdot g(y)} = \exp\left(-\frac{1}{2}y^2\right) \tag{6}$$

이 확률은 a또는 b와 무관하다.

지금까지 내용을 정리하면, 절단정규확률분포에서 난수를 생성하는 채택기각법은 다음과 같다.

(1단계) 지지대 (a, b)상에서 일양난수 y를 생성한다.

(2단계) 지지대 $(0, 1)$상에서 일양난수 u를 생성한다.

(3단계) 만약 식 $u < \exp\left(-\frac{1}{2}y^2\right)$이 성립하면 $x = y$로 하고, 그렇지 않으면 제1단계로 돌아간다.

만약 a의 절대값과 b가 아주 크지 않으면, 정규확률분포의 확률밀도함수 $f(x)$는 일양확률밀도함수에 가까우며, 따라서 이 알고리즘은 효율적이다.

채택기각법을 적용해서 지지대를 구간 $(-2, 2)$로 제한한 절단정규확률분포에서 난수들을 생성하기 위해서, 다음 MATLAB파일 AcceptRejectTruncatedNormal101.m을 실행하라.

```
% ------------------------------------------------------------------------
%  Filename: AcceptRejectTruncatedNormal101.m
%  Generated Truncated Normal Random Numbers
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all
rng((1001^2+1)/2,'twister')
N = 1000
% set up storage arrays
xacc = zeros(1,N);        % random number
yacc = zeros(1,N);        % corresponding y value
xrej = zeros(1,N);        % rejected random number
yrej = zeros(1,N);        % corresponding y value
iiacc = 1;
```

```
iirej = 1;
a = -2, b = 2
Na = normcdf(a), Nb = normcdf(b)
dumc = 1/(Nb-Na)/sqrt(2*pi)
c = dumc*(b-a)
% constant
while iiacc <= N
    xdum = (b-a)*rand(1)+a;
    u = rand(1);
    ff = dumc*exp(-1/2*xdum.^2);
    gg = 1/(b-a);
    if u <= ff/c/gg
        xacc(iiacc) = xdum;
        yacc(iiacc) = u*gg*c;
        iiacc = iiacc+1;
    else
        xrej(iirej) = xdum;
        yrej(iirej) = u*gg*c;
        iirej = iirej+1;
    end
end
% 2-D Plot of Accept-Reject Regions
xdum2 = linspace(a,b,501);
ydum2 = dumc*exp(-0.5*xdum2.^2);
ydum3 = 1/(b-a);
plot(xacc,yacc,'b.',xrej,yrej,'bO',xdum2,ydum2,'r', ...
     xdum2,ydum3,'k--','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
legend('\bf Accept','\bf Reject','\bf f(x)','\bf g(x)','location','NW')
saveas(gcf,'AcceptRejectTruncatedNormal101a','epsc')
% Goodness-of-Fit test
RN = xacc;                  % RN: Generated random numbers
edges = linspace(-2,2,41);
expectedCounts = N * diff(normcdf(edges))/(normcdf(2)-normcdf(-2))
[H,P_value,stat] = chi2gof(RN,'edges',edges, ...
                           'expected',expectedCounts)
% Plotting Histogram and True PDF
[ Nbar,xcenter ] = hist(xacc,-1.95:0.10:1.95)
Nheight = Nbar/( xcenter(2)-xcenter(1) )/N;
xx = linspace(a,b,1001);
pp = dumc*exp(-0.5*xx.^2);            % True PDF
figure
bar(xcenter,Nheight,1,'w')
set(gca,'fontsize',11,'fontweigh','bold')
hold on
plot(xx,pp,'r','linewidth',2)
xlabel('x'), ylabel('Relative Frequency')
hold off
axis([ -2.2 2.2 0 0.6 ])
saveas(gcf,'AcceptRejectTruncatedNormal101b','epsc')
save('AcceptRejectTruncatedNormal101','xacc','yacc')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면 알 수 있듯이, 카이제곱통계량은 40.5871이고 자유도는 39이며 p값이 0.4003으로 이 난수들이 이 절단정규확률분포로부터 생성되었다는 귀무가설이 채택된다. 이 MATLAB프로그램을 실행하면, 그림 1.4.3와 그림 1.4.4가 출력된다. 그림

1.4.3에서 점(dot)은 채택되는 점을 나타내고 작은 원은 기각된 점을 나타낸다. 그림 1.4.4에서 막대그래프는 이 난수들의 히스토그램이고, 적색 실선은 진짜확률밀도함수이다. 그림 1.4.4에서 이 난수들이 절단정규확률분포로부터 발생되었다는 것을 확인할 수 있다. ■

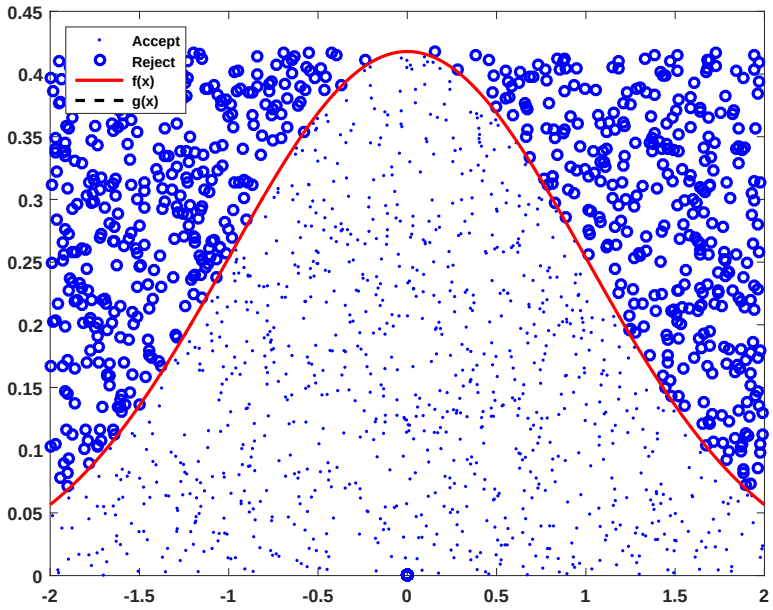


그림 1.4.3. 채택기각법과 절단정규확률분포

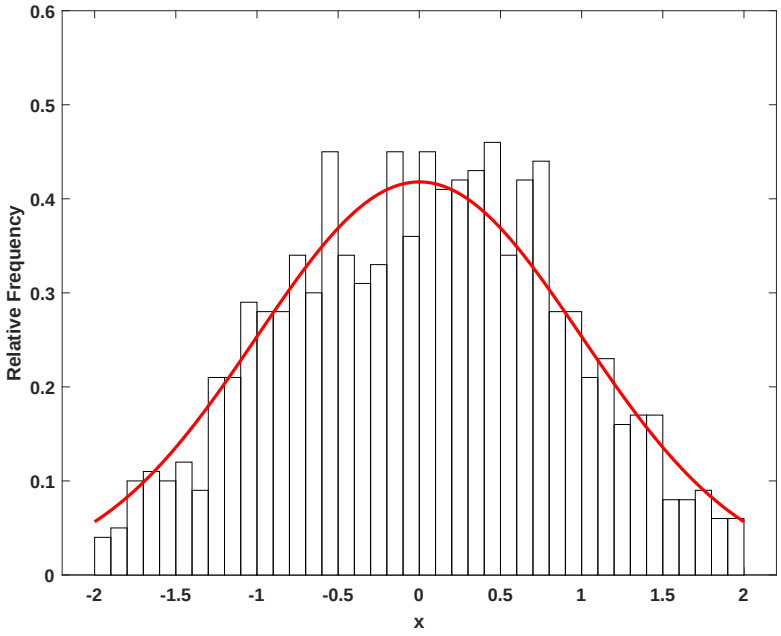


그림 1.4.4. 채택기각법과 절단정규확률분포의 히스토그램

채택기각법에서 확률밀도함수 $f(x)$로부터 난수 x를 하나 생성하기까지 낭비한 확률밀도함수 $g(x)$로부터 발생되는 난수 y의 개수는 모수가 $1/c$인 기하분포에 따른다. 즉, 어떤 y가

x로 채택될 확률은 $1/c$이고, 이 확률은 각 단계에 있어서 항상 일정하며, 각 단계에서 채택 여부는 서로 독립이다. 폐기되는 난수들의 개수를 확률변수 K로 나타내면, k개 난수들이 폐기되고 제 $k+1$번째 난수가 채용될 확률은 다음과 같다.

$$Pr(K=k) = \left[1-\frac{1}{c}\right]\left[\frac{1}{c}\right]^k, \qquad (k=0,1,\cdots) \tag{1.4.9}$$

즉, K는 모수가 파라메터 $1/c$인 기하확률분포를 따른다. 채택기각법에 의해 난수가 채용될 확률은 $1/c$이므로, 상수 c가 1에 가까우면 알고리즘 1.4.1의 효율이 좋다.

예제 1.4.3 채택기각법을 적용해서 정규난수들을 발생시켜보자. 표준정규확률변수 z에 대해서 절대정규확률변수 $x \doteq |z|$의 확률밀도함수는 다음과 같다.

$$f(x) = \sqrt{\frac{2}{\pi}}\exp\left(-\frac{1}{2}x^2\right) \quad (x \geq 0) \tag{1}$$

지수확률변수 y의 확률밀도함수 $g(y) = \exp(-y)$로부터 쉽게 난수를 발생시킬 수 있다. 즉, 일양난수 u에 대해서 $y = -\ln u$는 평균이 1인 지수난수이다. 다음 식들이 성립한다.

$$\frac{f(x)}{g(x)} = \sqrt{\frac{2}{\pi}}\exp\left(x-\frac{1}{2}x^2\right) \leq \sqrt{\frac{2e}{\pi}} \tag{2}$$

따라서, $2e/\pi$를 상수 c로 선택한다. 즉, 이 채택기각법에 의해 난수가 채용될 확률은 $1/c \approx 0.76$이다. 이렇게 생성된 절대정규난수들 $x_1, x_2, \cdots$로부터 정규난수들을 생성하기 위해서는, 새로운 일양난수 u_j를 발생시켜서 이 일양난수가 0.5보다 크면 $z_j = x_j$라고 두고 그렇지 않으면 $z_j = -x_j$로 둔다.

지금까지 설명한 방법을 적용해서 정규확률분포로부터 난수들을 생성하기 위해서, 다음 MATLAB프로그램 AcceptRejectNormal101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: AcceptRejectNormal101.m
%  Generating Normal RN by Acceptance-Rejection Method
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
tic;
rng(240841,'twister')
N = 100, N25 = N*2.5;
yy = - log(rand(1,N25)); % yy: random numbers from g(y) = exp(-y)
c = sqrt(2*exp(1)/pi), AccPr = 1/c
ff = sqrt(2/pi)*exp(-yy.^2/2);
gg = exp(-yy);
dum = (rand(1,N25) <= ff./gg/c);
```

```
[r c yabs] = find(yy.*dum);
yabs = yabs(1:N);                    % yabs: RN from |N(0,1)|
yran = sign(rand(1,N)-0.5).*yabs;    % yran: RN from N(0,1)
toc;
% Goodness-of-Fit Test by Lilliefors
[H,P_value,stat,CV] = lillietest(yran,0.05,'norm')
% Plotting Histogram and True PDF
[ Nbar,xcenter ] = hist(yran,21);
Nheight = Nbar/( xcenter(2)-xcenter(1) )/N;
bar(xcenter,Nheight,1,'w')
set(gca,'fontsize',11,'fontweigh','bold')
hold on
xx = linspace(-4,4,201);
pp = normpdf(xx);                    % True PDF
plot(xx,pp,'r','linewidth',2)
legend('\bf Histogram','\bf True PDF',1)
xlabel('x'), ylabel('Relative Frequency')
axis([-4 4 0 0.6])
hold off
saveas(gcf,'AcceptRejectNormal101','epsc')
save('AcceptRejectNormal101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 표준정규확률분포에서 100개 난수들이 생성된다. MATLAB함수 lillietest을 사용해서 Lilliefors적합성검정을 한 결과, 카이제곱통계량은 0.0740이고 임계값(critical value: CV)는 0.0890이며 p값이 0.1937이다. 따라서 이 난수들이 이 정규확률분포로부터 생성되었다는 귀무가설이 채택된다. 이 MATLAB프로그램을 실행하면, 그림 1.4.5가 출력된다. 그림 1.4.5에서 막대그래프는 이 난수들의 히스토그램이고, 적색 실선은 진짜확률밀도함수이다. 그러나, 이 방법에서는 정규난수 1개를 생성하기 위해서 일양난수를 적어도 3개 발생시켜야 하고, 또한 이 채택기각법에 의해 난수가 선택될 확률은 0.76은 1과 아주 가까운 값은 아니다. 따라서, 이 난수발생법은 정규난수를 생성하는 좋은 방법은 아니다. ■

채택기각법을 적용해서 지지대가 이산집합 S인 이산확률질량함수 $\{p_k \mid k \in S\}$에서 난수를 생성하기로 하자. 지지대가 S이고 난수를 생성하기가 쉬운 이산확률질량함수를 $\{q_k \mid k \in S\}$라 하자. 다음 식을 만족하는 상수 c를 선택하자.

$$\frac{p_k}{q_k} \le c \quad (k \in S) \tag{1.4.10}$$

이산확률질량함수 $\{p_k \mid k \in S\}$에서 난수 x를 발생시키기 위해서는, 먼저 이산확률질량함수 $\{q_k \mid k \in S\}$에서 난수 y를 생성하는 동시에 일양난수 u를 생성한다. 만약 u가 $p_y/[cq_y]$보다 작으면, 이 y를 이산확률질량함수 $\{p_k \mid k \in S\}$에서 생성된 난수 x로 채택한다. 만약 그렇지

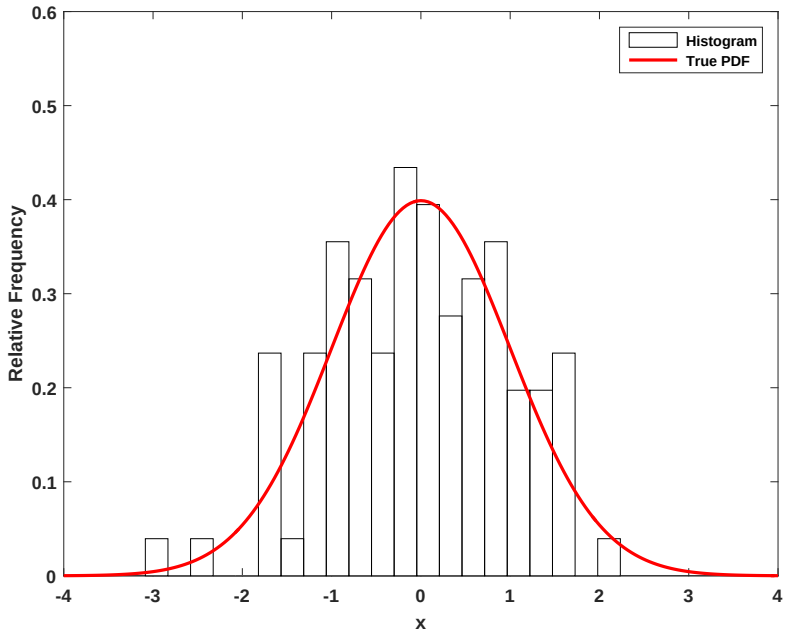


그림 1.4.5. 채택기각법과 정규확률분포

않으면, 다시 이산확률질량함수 $\{q_k \mid k \in S\}$에서 난수 y를 생성한다. 우선, 난수 $y = k$가 난수 x로 채택될 확률은 다음과 같다.

$$Pr(y = k, accept) = Pr(y = k)Pr(accept \mid y = k) = q_k \frac{p_k}{cq_k} = \frac{p_k}{c} \tag{1.4.11}$$

따라서, 후보값 y가 난수 x로 채택될 확률은 다음과 같다.

$$Pr(accept) = \sum_{k \in S} Pr(y = k, accept) = \sum_{k \in S} \frac{p_k}{c} = \frac{1}{c} \tag{1.4.12}$$

따라서 다음 식들이 성립한다.

$$Pr(x = k) = \sum_{n=1}^{\infty} \left[1 - \frac{1}{c}\right]^{n-1} \frac{p_k}{c} = p_k \tag{1.4.13}$$

식 (1.4.13)에서 알 수 있듯이, 위에 기술한 과정은 이산형 채택기각법을 이룬다.

예제 1.4.4 어떤 주사위의 눈이 도출될 확률이 다음과 같다고 하자.

$$p_1 = \frac{1}{21}, \quad p_2 = \frac{3}{21}, \quad p_3 = \frac{5}{21}, \quad p_4 = \frac{6}{21}, \quad p_5 = \frac{4}{21}, \quad p_6 = \frac{2}{21} \tag{1}$$

채택기각법을 적용해서, 이 주사위로부터 발생할 수 있는 난수들을 생성하기로 하자.

공정한 주사위의 확률질량함수는 다음과 같다.

$$q_i = \frac{1}{6}, \qquad (i = 1, 2, 3, 4, 5, 6) \tag{2}$$

따라서, 상수 c를 다음과 같이 선택하자.

$$c = \max_i \frac{p_i}{q_i} = \frac{6/21}{1/6} = \frac{12}{7} \tag{3}$$

만약 일양난수 u가 $p_y/[cq_y]$보다 작으면, 이 후보값 y를 이산확률질량함수 $\{p_i\}$에서 생성된 난수 x로 채택한다. 그렇지 않으면, 다시 이산확률질량함수 $\{q_i\}$에서 후보값 y를 생성한다.

지금까지 설명한 방법을 적용해서 이산형 확률분포로부터 난수들을 생성하기 위해서, 다음 MATLAB프로그램 AcceptRejectDiscrete101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: AcceptRejectDiscrete101.m
%  Generating Discrete Random Numbers 1
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
rng((1001^2+1)/2,'twister')
N = 1000
p = [ 1 3 5 2 4 6 ]/21
q =  ones(1,6)/6
for jj = 1:N
    kk = 0;
    while 12345
          ii = ceil(6*rand); kk = kk + 1;
          if rand < p(ii)*7/2      % c*q(ii) = 12/7*1/6 = 2/7
             yran(jj)=ii; cc(jj)=kk; break
          end
    end
end
cmean = mean(cc)            % averge number of steps until success
AcceptProb = 1/cmean        % acceptance probability
% Chi-squared Goodness-of-Fit test
edge = linspace(0.5,6.5,7);
expectedCount = N*p;
[H,pvalue,stat] = chi2gof(yran,'edges',edge,'expected',expectedCount)
% Histogram
yFreq = hist(yran,6)/N
bar(1:1:6,yFreq,0.1,'w')
set(gca,'fontsize',11,'fontweigh','bold')
hold on
plot(1:6,p,'r-*','linewidth',2)     % True PDF
legend('\bf Histogram','\bf True PDF',2)
xlabel('x'), ylabel('Relative Frequency')
axis([0 7 0 0.4 ])
hold off
saveas(gcf,'AcceptRejectDiscrete101','epsc')
save('AcceptRejectDiscrete101','yran')
% End of program
```

```
39 % ------------------------------------------------------------
```

이 MATLAB 프로그램을 실행하면, 확률질량함수 $\{p_k\}$ 에서 100 개 난수들을 생성한다. 채택기각법을 사용해서 확률질량함수 $\{p_k\}$ 로부터 난수 1 개를 생성하기 위해서, 일양확률질량함수 $\{q_k\}$ 에서 평균 $\hat{c} = 1.6460$ 개의 난수들을 생성한다. 즉, 채택률은 0.6075 이다. 적합도를 검정하기 위해서 카이제곱검정을 한 결과, 카이제곱통계량값은 0.9534 이고 자유도는 5 이며 p 값은 0.9662 이다. 즉, 이 난수들이 확률질량함수 $\{p_k\}$ 에서 발생되었다는 귀무가설을 채택한다. 이렇게 발생시킨 난수들의 히스토그램이 그림 1.4.6에 그려져 있다. 그림 1.4.6에서 막대그래프는 이 난수들의 히스토그램이고, 적색 별표는 진짜확률질량함수이다. 그림 1.4.6에서 이 난수들이 절단정규확률분포로부터 발생되었다는 것을 확인할 수 있다. ■

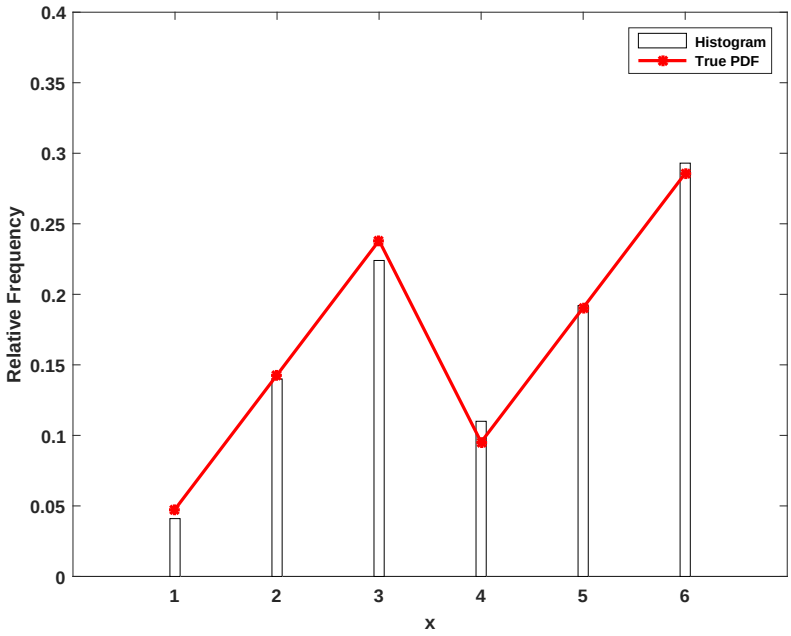


그림 1.4.6. 채택기각법과 이산형 확률분포

제1.5절 연속형 확률분포

이 절에서는 자주 사용되는 연속형 확률분포에서 난수를 발생시키는 방법에 대해서 살펴보자.

1.5.1 일양확률분포

지지대가 (a,b)인 일양확률분포 $U(a,b)$의 확률밀도함수는 다음과 같다.

$$f(x) = \frac{1}{b-a} 1_{(a,b)}(x) \tag{1.5.1}$$

또한, 누적확률분포함수 $F(x)$는 다음과 같다.

$$F(x) = \frac{x-a}{b-a} 1_{(a,b)}(x) + 1_{[b,\infty)}(x) \tag{1.5.2}$$

이 일양확률분포의 평균과 분산은 각각 다음과 같다.

$$E(x) = \frac{a+b}{2}, \quad Var(x) = \frac{[b-a]^2}{12} \tag{1.5.3}$$

지지대가 $(0,1)$인 일양확률변수 u에 대해서 다음 식이 성립함은 명백하다.

$$a + [b-a]u \quad \overset{d}{\sim} \quad U(a,b) \tag{1.5.4}$$

식 (1.5.4)를 사용해서 지지대가 (a,b)인 일양난수들을 발생시킨다.

예제 1.5.1 MATLAB을 이용해서 일양확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 일양난수를 살펴보기 위해서, 다음 MATLAB프로그램 ContinuousUniformRV101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: ContinuousUniformRV101.m
%  Contiuous Uniform Random Variable
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
a = -1, b =2                          % Parameters
% PDF
yp1 = pdf('unif',1,a,b)
yp2 = unifpdf(1,a,b)
xx = a:0.05:b;
yy = pdf('Uniform',xx,a,b);
subplot(2,2,1)
plot(xx,yy,'r-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
title('\bf PDF')
% CDF
yc1 = cdf('unif',1,a,b)
yc2 = unifcdf(1,a,b)
xx = a:0.05:b;
```

```
yy = cdf('Uniform',xx,a,b);
subplot(2,2,2)
plot(xx,yy,'g-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
title('\bf CDF')
% Inverse CDF
yi1 = icdf('unif',0.5,a,b)
yi2 = unifinv(0.5,a,b)
xx = 0:0.02:1;
yy = icdf('Uniform',xx,a,b);
subplot(2,2,3)
plot(xx,yy,'b-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
title('\bfInverse CDF')
% Random Numbers
yr1 = random('unif',a,b,3)
yr2 = unifrnd(a,b,3)
rand('twister',5489)
yran = random('Uniform',a,b,1,100);
subplot(2,2,4)
h1 = histogram(yran)
h1.BinWidth = 0.25;
h1.FaceColor = [1 1 1];
h1.EdgeColor = 'black';
set(gca,'fontsize',11,'fontweigh','bold')
title('\bf Histogram')
saveas(gcf,'ContinuousUniformRV101','epsc')
save('ContinuousUniformRV101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 지지대가 $(-1, 2)$ 인 일양확률변수의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 100 개 일양난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.1에 그려져 있다.

연속형 일양확률분포에 관한 MATLAB함수들로는 unifpdf, pdf, unifcdf, cdf, unifinv, icdf, unifstat, unifit, mle, unifrnd rand, random 등이 있다. ■

예제 1.5.2 R을 이용해서 일양확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 일양난수를 살펴보기 위해서, 다음 R프로그램 ContinuousUniformRV101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: ContinuousUniformRV101R.R
# Continuous Uniform Random Variable
# Programmed by CBS
# ------------------------------------------------------------------------
a <- -1; b <- 2
nAbsissa <- 301
x <- seq(a,b,len=nAbsissa)
```

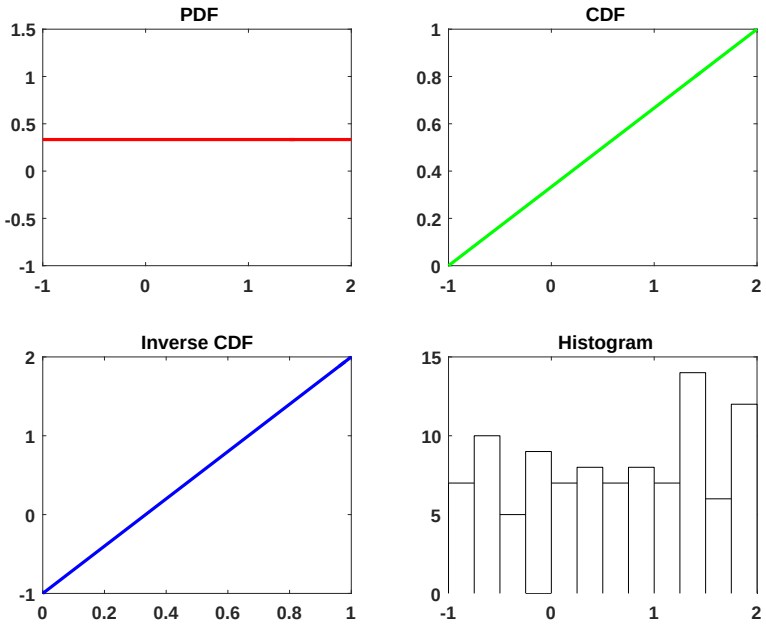


그림 1.5.1. 일양확률변수와 MATLAB

```
yp <- dunif(x, min=a, max=b, log=FALSE)   # PDF
yc <- punif(x, min=a, max=b, lower.tail=TRUE, log.p=FALSE)  # CDF
ix <- seq(0,1,len=nAbsissa)
yi <- qunif(ix, min=a, max=b, lower.tail=TRUE, log.p=FALSE) # Inverse CDF
nSim <- 100
set.seed(41)
yran <- runif(nSim, min=a, max=b)

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('ContinuousUniformRV101R.eps')  # Start to save figure
RVdata1 <- data.frame(x,yp,yc)
RVdata2 <- data.frame(ix,yi)
RVdata3 <- data.frame(yran)
plot11 <- ggplot(RVdata1, aes(x,yp)) +
          geom_line(col="red",lwd=1.2) +
          xlab("x") +
          ggtitle("PDF")
plot12 <- ggplot(RVdata1, aes(x,yc)) +
          geom_line(col="dark green",lwd=1.2) +
          xlab("x") +
          ggtitle("CDF")
plot13 <- ggplot(RVdata2, aes(ix,yi)) +
          geom_line(col="blue",lwd=1.2) +
          xlab("x") +
          ggtitle("Inverse CDF")
plot14 <- ggplot(RVdata3, aes(x=yran)) +
          geom_histogram(binwidth=0.25,fill="white",color="black")+
          xlab("x") +
          ggtitle("Histogram")
pushViewport(viewport(layout=grid.layout(2,2)))
```

```
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot13, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot14, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()             # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 지지대가 $(-1, 2)$인 일양확률변수의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 일양확률분포에서 100개 일양난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.2에 그려져 있다. ■

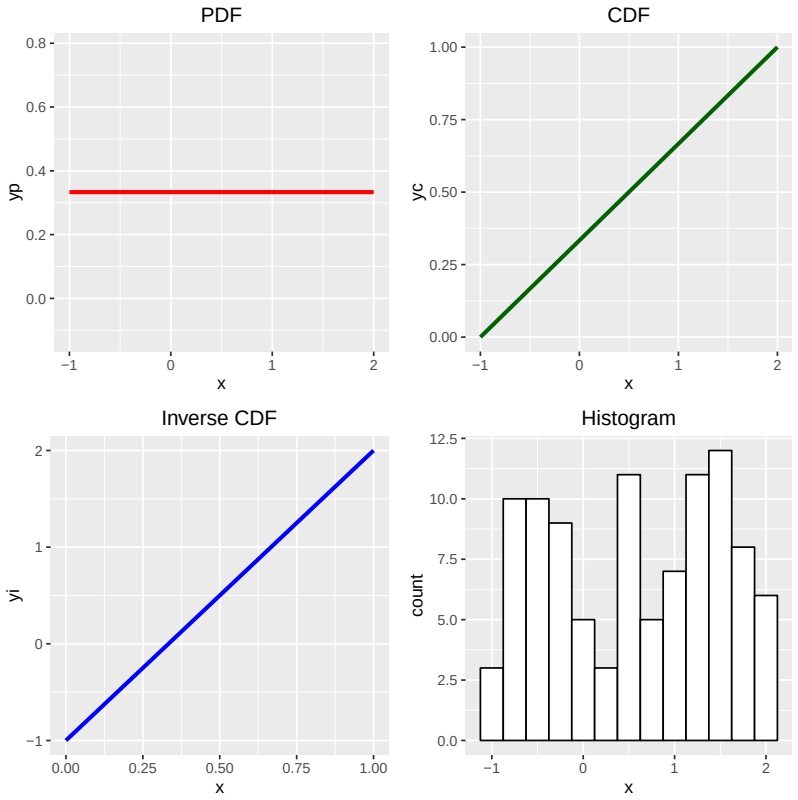


그림 1.5.2. 일양확률변수와 R

1.5.2 정규확률분포

Gauss확률분포라고도 불리우는 정규확률분포는 1733년 A. De Moivre가 처음 발견했고, 1809년에 C. F. Gauss가 그리고 1812년에 P. S. Laplace에 의해서 재발견되었다.

모수벡터가 $[\mu, \sigma]$인 정규확률분포의 확률밀도함수는 다음과 같다.

$$n(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{[x-\mu]^2}{2\sigma^2}\right) \tag{1.5.5}$$

이 정규확률분포 $\mathcal{N}(\mu, \sigma^2)$의 누적확률분포함수를 $N(x; \mu, \sigma^2)$로 표기하자. 이 정규확률분포의 평균은 μ이고 분산은 σ^2이다. 평균이 0이고 분산이 1인 정규확률분포를 표준정규확률분포라 하고 확률밀도함수와 누적확률분포함수를 각각 $n(0,1)$와 $N(0,1)$으로 표기한다.

일양난수에서 정규난수(normal random number)를 생성하는 첫 번째 방법은 중심극한정리를 이용하는 것이다. 지지대가 $(0,1)$인 일양확률변수들 $\{u_j\}$에 대해서 다음 식이 성립한다.

$$\frac{\sum_{j=1}^{n} u_j - \frac{n}{2}}{\sqrt{\frac{n}{12}}} \xrightarrow{d} \mathcal{N}(0,1) \tag{1.5.6}$$

여기서 ' $\xrightarrow{d}$ '는 분포수렴을 의미한다. 정규난수를 생성하기 위해서는 보통 $n = 12$를 사용한다.

두 번째 방법은 Box-Müller법이다. 구간 $(0,1)$상에서 일양확률분포를 따르는 확률변수 u_1에 대해서, 확률벡터 $[\cos(2\pi u_1), \sin(2\pi u_1)]$은 단위원 상에서 일양확률분포를 따른다. 확률변수 u_1과 독립이고 다음과 같은 확률분포를 따르는 확률변수 ρ를 생각해보자.

$$Pr(\rho \le x) = \int_0^x r \exp\left(-\frac{1}{2}r^2\right) dr = 1 - \exp\left(-\frac{1}{2}x^2\right) \tag{1.5.7}$$

다음 확률변수들을 정의하자.

$$x \doteq \rho\cos(2\pi u_1), \quad y \doteq \rho\sin(2\pi u_1) \tag{1.5.8}$$

다음 식들이 성립한다.

$$\rho = \sqrt{x^2 + y^2} \quad u_1 = \frac{1}{2\pi}\arctan\frac{y}{x} \tag{1.5.9}$$

Jacobian은 다음과 같다.

$$J = \left|\frac{\partial(\rho, u_1)}{\partial(x,y)}\right| = \begin{vmatrix} \frac{x}{\sqrt{x^2+y^2}} & \frac{y}{\sqrt{x^2+y^2}} \\ \frac{-y}{2\pi[x^2+y^2]} & \frac{x}{2\pi[x^2+y^2]} \end{vmatrix} = \frac{1}{2\pi\rho} \tag{1.5.10}$$

식 (1.5.9)와 식 (1.5.10)에서 알 수 있듯이, 다음 식들이 성립한다.

$$\rho\exp\left(-\frac{1}{2}\rho^2\right) d\rho du_1 = \frac{1}{2\pi}\exp\left(-\frac{1}{2}[x^2+y^2]\right) dxdy = n(x)dx\, n(y)dy \tag{1.5.11}$$

즉, 확률변수들 x와 y는 표준정규확률분포를 따른다. 만약 u_2가 구간 $(0,1)$상에서 일양확

률분포를 따르면, $\sqrt{-2\ln u_2}$는 식 (1.5.7)의 확률분포를 따른다. 따라서, 다음과 같은 결론을 얻는다. 서로 독립이고 구간 $(0,1)$상에서 일양확률분포를 따르는 확률변수들 u_1과 u_2에 대해서, 다음 확률변수들을 정의하자.

$$z_1 \doteq \sqrt{-2\ln u_2}\cos(2\pi u_1), \quad z_2 \doteq \sqrt{-2\ln u_2}\sin(2\pi u_1) \tag{1.5.12}$$

이 z_1과 z_2는 서로 독립인 표준정규확률변수들이다. 이 결론을 바탕으로 정규난수들을 생성하는 Box-Müller법은 로그함수와 삼각함수의 계산에서 시간이 걸린다는 단점을 지니고 있다. 앞에서도 언급했듯이, 다음 승산선형합동생성기는 완전주기를 갖는다.

$$x_n = 16807x_{n-1} \pmod{2^{31}-1}, \quad (n = 1, 2, \cdots) \tag{1.5.13}$$

그러나, 이렇게 생성된 일양난수들에 Box-Müller법을 적용하면, 가능한 가장 작은 정규난수는 -4.476근처이다. 이론상 2.15×10^9개의 정규난수들 중에서 -4.476보다 작은 값이 약 8000개 나와야 한다.

예제 1.5.3 잘못 생성된 일양난수들을 사용하면 Box-Müller법도 적절한 방법이 되지 못함을 보이기 위해서, 다음 MATLAB프로그램 BoxMuller_Nonnormality101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: BoxMuller_Nonnormality101.m
%  Nonnormality of Box-Muller Transformation
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
a = 2^17;
N = 5000
N05 = N/2
v(1) = 12345;
for jj=2:1:N
    v(jj) = mod(97*v(jj-1),a);
end
u = v/a;
u1 = u(1:2:end);
u2 = u(2:2:end);
r = sqrt(-2*log(u1));
theta = 2*pi*u2;
x = r.*cos(theta);
y = r.*sin(theta);
% Plotting
plot(x,y,'r.')
set(gca,'fontsize',11,'fontweigh','bold')
axis([-4 4 -4 4 ])
axis square
saveas(gcf,'BoxMuller_Nonnormality101','epsc')
save('BoxMuller_Nonnormality101','x','y')
```

```
% End of program
% --------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 다음과 같은 승산선형합동생성기로부터 일양난수들을 생성한다.

$$v_n = 97v_{n-1} \pmod{2^{17}} \quad (n = 1, 2, \cdots) \tag{1}$$

이렇게 생성된 일양난수들에 Box-Müller변환을 적용해서 얻은 정규난수들의 산점도가 그림 1.5.3에 그려져 있다. 그림 1.5.3에서 알 수 있듯이, 이 Box-Müller법에 의한 난수들이 서로 독립인 정규난수들이라고 말할 수는 없다. ■

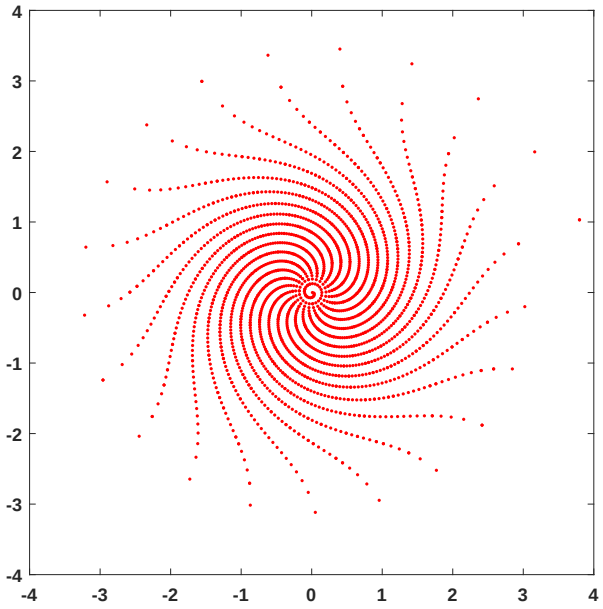


그림 1.5.3. Box-Müller법과 비정규난수

세 번째 방법은 Box-Müller법을 약간 변형한 것으로서 Marsaglia & Bray [31]에 의해서 제시되었다. 서로 독립이고 구간 $(0, 1)$상에서 일양확률분포를 따르는 확률변수들 u_1과 u_2에 대해서, 다음 확률변수들을 정의하자.

$$v_1 \doteq 2u_1 - 1, \quad v_2 \doteq 2u_2 - 1 \tag{1.5.14}$$

확률벡터 $[v_1, v_2]$의 극좌표를 $[\rho, \theta]$라고 하면, 다음 식들이 성립한다.

$$v_1 = \rho cos\theta, \quad v_2 = \rho sin\theta \tag{1.5.15}$$

확률벡터 $[v_1, v_2]$의 지지대를 $\{0 \le \rho \le 1\}$로 제한하면, 다음 식들이 성립한다.

$$Pr\left(\rho^2 \le a\right) = Pr\left(\rho \le \sqrt{a}\right) = \frac{\pi\sqrt{a^2}}{\pi} = a \tag{1.5.16}$$

식 (1.5.16)에서 알 수 있듯이, 다음 식이 성립한다.

$$\rho^2 \overset{d}{\sim} U(0,1) \tag{1.5.17}$$

또한, 다음 식이 성립함을 쉽게 알 수 있다.

$$\frac{1}{2\pi}\theta \overset{d}{\sim} U(0,1) \tag{1.5.18}$$

또한, 확률변수들 ρ^2와 θ는 서로 독립이다. 다음 확률변수들을 정의하자.

$$z_3 \doteq \sqrt{-2\ln\rho^2}\,\frac{v_1}{\rho}, \quad z_4 \doteq \sqrt{-2\ln\rho^2}\,\frac{v_2}{\rho} \tag{1.5.19}$$

Box-Müller법에서 알 수 있듯이, z_3과 z_4는 서로 독립인 표준정규확률변수들이다. 식 (1.5.19)를 사용해서 정규난수를 생성하는 방법을 Box-Müller-Marsaglia법 또는 극좌표법(polar method)이라 부른다. 이 방법은 두 단계들로 구성된다. 첫째, 지지대가 $(-1,1)$인 일양난수들 v_1과 v_2를 생성한다. 둘째, 만약 $w = v_1^2 + v_2^2$가 식 $w \ge 1$을 만족하면, 다시 일양난수들 v_1과 v_2를 생성한다. 그렇지 않은 경우에 $c = \sqrt{-[2\ln w]/w}$를 계산하면, cv_1과 cv_2는 서로 독립인 정규난수들이다.

예제 1.5.4 Box-Müller-Marsaglia법을 사용해서 정규난수들을 생성하기 위해서 다음 MATLAB프로그램 BoxMullerMarsaglia101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: BoxMullerMarsaglia101.m
%  Generating Normal RN by Box-Muller_Marsaglia 1
%  Programmed by CBS
%-------------------------------------------------------------------
clear all, close all
rand('twister',5489)
N = 100, N25 = N*2.5;
v1 = 2*rand(1,N25)-1;
v2 = 2*rand(1,N25)-1;
w = v1.^2 + v2.^2;
wIndex = find(w <= 1);
wIndex = wIndex(1:N);
ww = w(wIndex);
vv1 = v1(wIndex);
```

```
vv2 = v2(wIndex);
c = sqrt( - 2*log(ww)./ww );
z1 = c.*vv1;
z2 = c.*vv2;
yran = [ z1 z2 ];
N2 = 2*N;                          % Number of Generated RN
% Goodness-of-Fit Test by Lilliefors
[H,P_value,stat,CV] = kstest(yran,[],0.05)
% Plotting Histogram and True PDF
xnum = -3.9:0.20:3.9;
N2height = hist(yran,xnum)/0.20/N2;
bar(xnum,N2height,1,'w')
set(gca,'fontsize',11,'fontweigh','bold')
hold on
xx = linspace(-4,4,201);
pp = normpdf(xx);                  % True PDF
plot(xx,pp,'r','linewidth',2)
legend('\bf Histogram','\bf True PDF',1)
xlabel('x'), ylabel('Relative Frequency')
axis([-4 4 0 0.65 ])
hold off
saveas(gcf,'BoxMullerMarsaglia101','epsc')
save('BoxMullerMarsaglia101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 표준정규난수들 $2 \times 100 = 200$개를 생성한다. 이 난수들이 표준정규확률분포에서 발생되었는지를 검정하기 위해서 Kolmogorov-Smirnov 검정을 한 결과, 검정통계량값은 0.0640이고 임계값은 0.0952이며 p값은 0.3703이다. 즉, 이 난수들이 표준정규확률분포에서 발생되었다는 귀무가설을 채택한다. 이 정규난수들의 히스토그램이 그림 1.5.4에 그려져 있다. 그림 1.5.4에서 이 난수들이 표준정규확률분포로부터 발생되었다는 것을 확인할 수 있다. ■

예제 1.5.5 MATLAB을 이용해서 정규확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 정규난수를 살펴보기 위해서, 다음 MATLAB프로그램 NormalRV101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: NormalRV101.m
%  Normal Random Variable
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
mu = 1.1, sigma = 0.8                    % Parameters
% PDF
yp1 = pdf('norm',1,mu,sigma)
yp2 = normpdf(1,mu,sigma)
xx = -3:0.01:5;
yy = pdf('Normal',xx,mu,sigma);
subplot(2,2,1)
plot(xx,yy,'r-','linewidth',2)
```

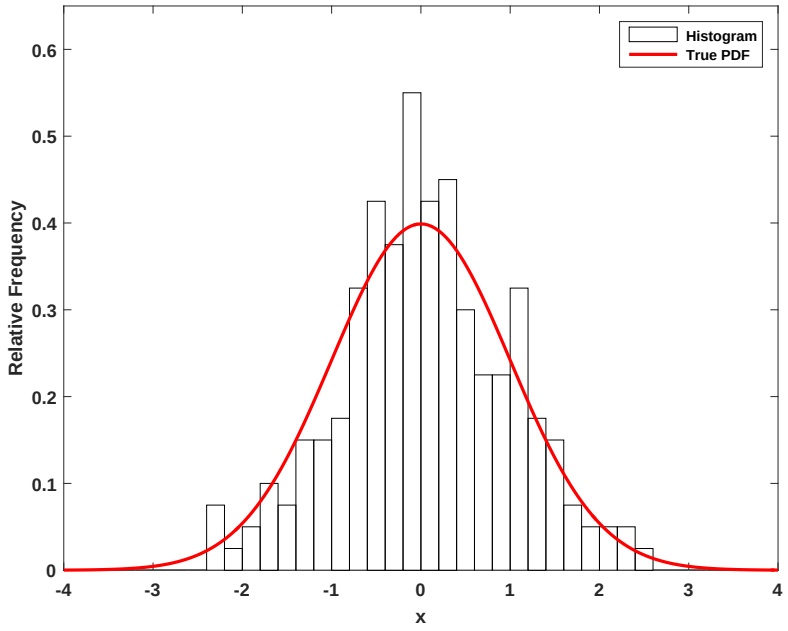


그림 1.5.4. Box-Müller-Marsaglia법

```
set(gca,'fontsize',11,'fontweigh','bold','xlim',[-3 5])
title('\bf PDF')
% CDF
yc1 = cdf('norm',1,mu,sigma)
yc2 = normcdf(1,mu,sigma)
xx = -3:0.01:5;
yy = cdf('Normal',xx,mu,sigma);
subplot(2,2,2)
plot(xx,yy,'g-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[-3 5])
title('\bf CDF')
% Inverse CDF
yi1 = icdf('norm',0.5,mu,sigma)
yi2 = norminv(0.5,mu,sigma)
xx = 0:0.01:1;
yy = icdf('Normal',xx,mu,sigma);
subplot(2,2,3)
plot(xx,yy,'b-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[-1.2 3.2])
title('\bfInverse CDF')
% Random Numbers
yr1 = random('norm',mu,sigma,3)
yr1 = normrnd(mu,sigma,3)
rand('twister',5489)
yran = random('Normal',mu,sigma,1,100);
subplot(2,2,4)
h1 = histogram(yran)
h1.BinWidth = 0.25;
h1.FaceColor = [1 1 1];
h1.EdgeColor = 'black';
set(gca,'fontsize',11,'fontweigh','bold')
title('\bf Histogram')
saveas(gcf,'NormalRV101','epsc')
save('NormalRV101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 정규확률분포 $\mathcal{N}(1.1., 0.9^2)$의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수가 그려진다. 또한, 이 확률분포에서 100개 정규난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.5에 그려져 있다.

정규확률분포에 관한 MATLAB함수들로는 normpdf, pdf, normcdf, cdf, norminv, icdf, normstat, normfit, mle, fitdist, dfittool, normlike, normrnd, randn, random 등이 있다. ■

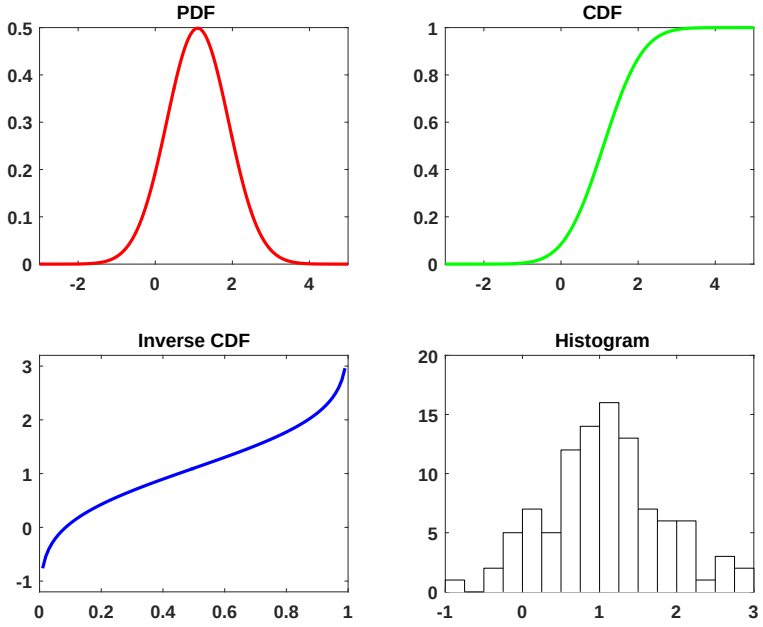


그림 1.5.5. 정규확률변수와 MATLAB

예제 1.5.6 R을 이용해서 정규확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 정규난수를 살펴보기 위해서, 다음 R프로그램 NormalRV101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: NormalRV101R.R
# Normal Random Variable
# Programmed by CBS
# ------------------------------------------------------------------------
nAbsissa <- 801
x <- seq(-3,5,len=nAbsissa)
yp <- dnorm(x, mean=1.1, sd=0.8, log=FALSE)   # PDF
yc <- pnorm(x, mean=1.1, sd=0.8, lower.tail=TRUE, log.p=FALSE)  # CDF
ix <- seq(0,1,len=nAbsissa)
yi <- qnorm(ix, mean=1.1, sd=0.8,lower.tail=TRUE, log.p=FALSE) #Inverse CDF
nSim <- 100
set.seed(41)
yran <- rnorm(nSim, mean=1.1, sd=0.8)

# Plotting
# install.packages("ggplot2")
library(ggplot2)
```

```
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('NormalRV101R.eps')  # Start to save figure
RVdata1 <- data.frame(x,yp,yc)
RVdata2 <- data.frame(ix,yi)
RVdata3 <- data.frame(yran)
plot11 <- ggplot(RVdata1, aes(x,yp)) +
          geom_line(col="red",lwd=1.2) +
          xlab("x") +
          ggtitle("PDF")
plot12 <- ggplot(RVdata1, aes(x,yc)) +
          geom_line(col="dark green",lwd=1.2) +
          xlab("x") +
          ggtitle("CDF")
plot13 <- ggplot(RVdata2, aes(ix,yi)) +
          geom_line(col="blue",lwd=1.2) +
          xlab("x") +
          ggtitle("Inverse CDF")
plot14 <- ggplot(RVdata3, aes(x=yran)) +
          geom_histogram(binwidth=0.25,fill="white",color="black")+
          xlab("x") +
          ggtitle("Histogram")
pushViewport(viewport(layout=grid.layout(2,2)))
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot13, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot14, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()            # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 정규확률분포 $\mathcal{N}(1.1., 0.9^2)$의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수가 그려진다. 또한, 이 확률분포에서 100개 정규난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.6에 그려져 있다. ■

1.5.3 대수정규확률분포

확률변수 $\ln x$가 정규확률분포를 따를 때, 확률변수 x는 대수정규확률분포(log normal distribution)를 따른다고 한다. 모수벡터가 $[\mu, \sigma]$인 대수정규확률분포의 확률밀도함수는 다음과 같다.

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma x} \exp\left(-\frac{[\ln x - \mu]^2}{2\sigma^2}\right) \tag{1.5.20}$$

이 대수정규확률분포의 평균과 분산은 각각 다음과 같다.

$$E(x) = \exp\left(\mu + \frac{\sigma^2}{2}\right), \quad Var(x) = \exp\left(2\mu + \sigma^2\right)\left[\exp(\sigma^2) - 1\right] \tag{1.5.21}$$

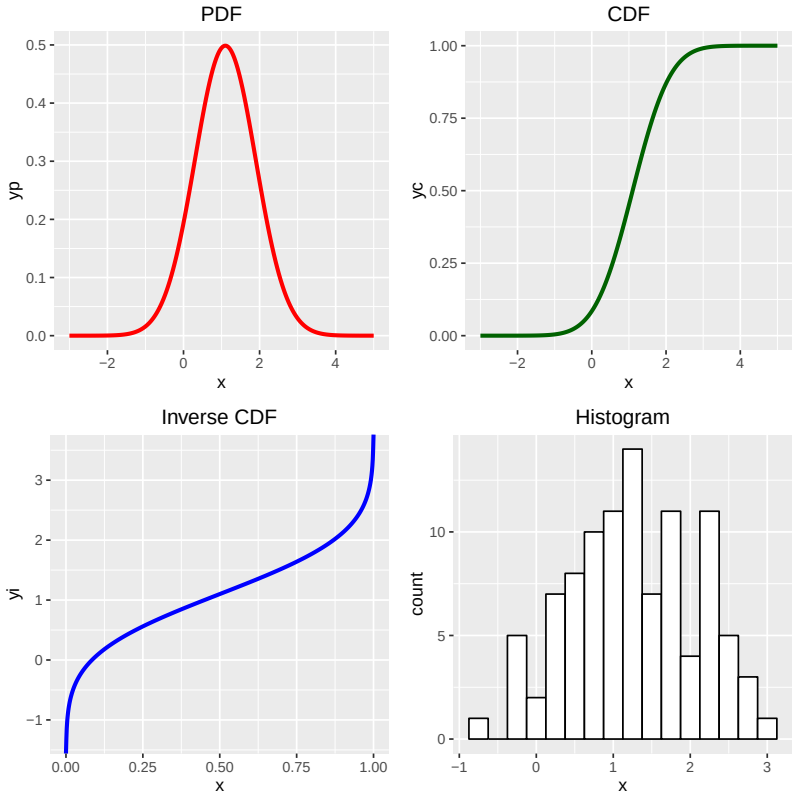


그림 1.5.6. 정규확률변수와 R

유의할 점은 μ와 σ^2은 각각 $\ln x$의 평균과 분산이라는 것이다.

정의에서 알 수 있듯이, 정규확률분포 $\mathcal{N}(\mu, \sigma^2)$에서 발생시킨 정규난수들 $\{v_j\}$에 대해서 $\{\exp(v_j)\}$는 모수벡터가 $[\mu, \sigma]$인 대수정규확률분포에서 발생된 난수들이다.

예제 1.5.7 MATLAB을 이용해서 대수정규확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 대수정규난수를 살펴보기 위해서, 다음 MATLAB프로그램 LogNormalRV101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: LognormalRV101.m
%  Lognormal Random Variable
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all
mu = 1, sigma = 1.1                          % Parameters
% PDF
yp1 = pdf('logn',1,mu,sigma)
yp2 = lognpdf(1,mu,sigma)
xx = 0:0.05:30;
yy = pdf('Lognormal',xx,mu,sigma);
subplot(2,2,1)
plot(xx,yy,'r-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold', ...
    'xlim',[0 30],'ylim',[0 0.3])
title('\bf PDF')
```

```
% CDF
yc1 = cdf('logn',1,mu,sigma)
yc2 = logncdf(1,mu,sigma)
xx = 0:0.05:30;
yy = cdf('Lognormal',xx,mu,sigma);
subplot(2,2,2)
plot(xx,yy,'g-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 30],'ylim',[0 1])
title('\bf CDF')
% Inverse CDF
yi1 = icdf('logn',0.5,mu,sigma)
yi2 = logninv(0.5,mu,sigma)
xx = 0:0.01:1;
yy = icdf('Lognormal',xx,mu,sigma);
subplot(2,2,3)
plot(xx,yy,'b-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[0 30])
title('\bfInverse CDF')
% Random Numbers
yr1 = random('logn',mu,sigma,3)
yr1 = lognrnd(mu,sigma,3)
rand('twister',5489)
yran = random('Lognormal',mu,sigma,1,500);
subplot(2,2,4)
h1 = histogram(yran)
h1.NumBins = 20;
h1.FaceColor = [1 1 1];
h1.EdgeColor = 'black';
set(gca,'fontsize',11,'fontweigh','bold')
title('\bf Histogram')
saveas(gcf,'LognormalRV101','epsc')
save('LognormalRV101','yran')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 모수들이 $\mu = 1$이고 $\sigma = 1.1$인 대수정규확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 100개 대수정규난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.7에 그려져 있다.

대수정규확률분포에 관한 MATLAB함수들로는 lognpdf, pdf, logncdf, cdf, logninv, icdf, lognstat, lognfit, mle, fitdist, dfittool, lognlike, lognrnd, random, randtool 등이 있다. ■

예제 1.5.8 R을 이용해서 대수정규확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 정규난수를 살펴보기 위해서, 다음 R프로그램 NormalRV101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: LogNormalRV101R.R
# Log Normal Random Variable
# Programmed by CBS
```

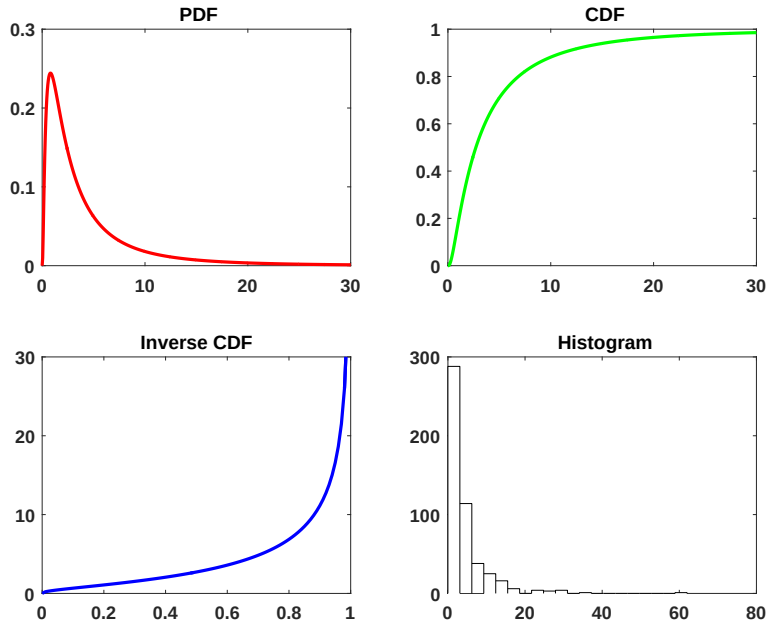


그림 1.5.7. 대수정규확률변수

```
# ------------------------------------------------------------------------
nAbsissa <- 801
x <- seq(0,30,len=nAbsissa)
yp <- dlnorm(x, meanlog = 1, sdlog = 1.1, log=FALSE)   # PDF
yc <- plnorm(x, meanlog = 1, sdlog = 1.1, lower.tail=TRUE, log.p=FALSE)  # CDF
ix <- seq(0,1,len=nAbsissa)
yi <- qlnorm(ix, meanlog = 1, sdlog = 1.1, lower.tail=TRUE, log.p=FALSE) #Inverse
    CDF
nSim <- 500
set.seed(41)
yran <- rlnorm(nSim, meanlog = 1, sdlog = 1.1)

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('LogNormalRV101R.eps')  # Start to save figure
RVdata1 <- data.frame(x,yp,yc)
RVdata2 <- data.frame(ix,yi)
RVdata3 <- data.frame(yran)
plot11 <- ggplot(RVdata1, aes(x,yp)) +
          geom_line(col="red",lwd=1.2) +
          xlab("x") +
          ggtitle("PDF")
plot12 <- ggplot(RVdata1, aes(x,yc)) +
          geom_line(col="dark green",lwd=1.2) +
          xlab("x") +
          ggtitle("CDF")
plot13 <- ggplot(RVdata2, aes(ix,yi)) +
          geom_line(col="blue",lwd=1.2) +
          xlab("x") +
          ggtitle("Inverse CDF")
plot14 <- ggplot(RVdata3, aes(x=yran)) +
```

```
        geom_histogram(bins=20,fill="white",color="black")+
        xlab("x") +
        ggtitle("Histogram")
pushViewport(viewport(layout=grid.layout(2,2)))
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot13, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot14, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()            # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 모수들이 $\mu = 1$이고 $\sigma = 1.1$인 대수정규확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 100개 대수정규난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.8에 그려져 있다. ■

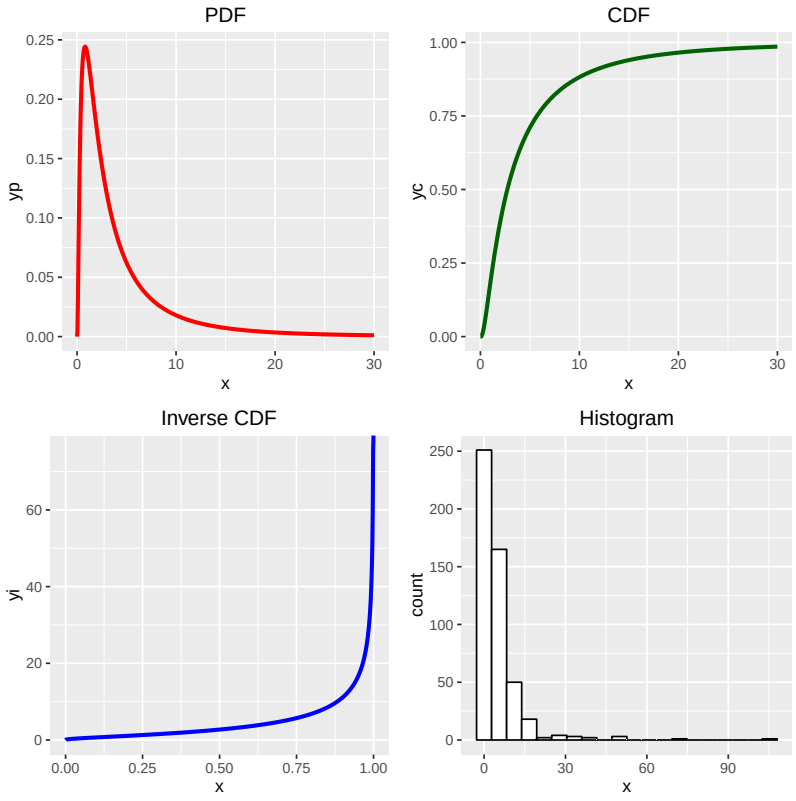


그림 1.5.8. 대수정규확률변수와 R

1.5.4 지수확률분포

비율모수(rate parameter)가 λ인 지수확률분포(exponential probability distribution)의 확률밀도함수는 다음과 같다.

$$f(x) = \lambda \exp(-\lambda x)\, 1_{[0,\infty)}(x) \tag{1.5.22}$$

이 지수확률분포의 누적확률분포함수는 다음과 같다.

$$F(x) = [1 - \exp(-\lambda x)]\, 1_{[0,\infty)}(x) \tag{1.5.23}$$

이 지수확률분포의 평균과 분산은 각각 다음과 같다.

$$E(x) = \frac{1}{\lambda}, \quad Var(x) = \frac{1}{\lambda^2} \tag{1.5.24}$$

또한, 적률모함수(moment generating function)은 다음과 같다.

$$m_x(t) = \left[1 - \frac{t}{\lambda}\right]^{-1} \qquad (t < \lambda) \tag{1.5.25}$$

지수확률분포은 대기행렬모형(queueing model)에서 자주 사용된다.

예제 1.5.9 MATLAB을 이용해서 지수확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 지수난수를 살펴보기 위해서, 다음 MATLAB프로그램 ExponentialRV101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: ExponentialRV101.m
%  Exponential Random Variable
%  Programmed by CBS
%-------------------------------------------------------------------------
clear all, close all
lambda = 0.5, mu = 1/lambda                                % Parameters
% PDF
yp1 = pdf('exp',1,mu)
yp2 = exppdf(1,mu)
xx = 0:0.05:10;
yy = pdf('Exponential',xx,mu);
subplot(2,2,1)
plot(xx,yy,'r-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 10],'ylim',[0 0.6])
title('\bf PDF')
% CDF
yc1 = cdf('exp',1,mu)
```

```
yc2 = expcdf(1,mu)
xx = 0:0.05:10;
yy = cdf('Exponential',xx,mu);
subplot(2,2,2)
plot(xx,yy,'g-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 10],'ylim',[0 1])
title('\bf CDF')
% Inverse CDF
yi1 = icdf('exp',0.5,mu)
yi2 = expinv(0.5,mu)
xx = 0:0.01:1;
yy = icdf('Exponential',xx,mu);
subplot(2,2,3)
plot(xx,yy,'b-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[0 10])
title('\bfInverse CDF')
% Random Numbers
yr1 = random('exp',mu,3)
yr1 = exprnd(mu,3)
rand('twister',5489)
yran = random('Exponential',mu,1,500);
subplot(2,2,4)
h1 = histogram(yran)
h1.NumBins = 20;
h1.FaceColor = [1 1 1];
h1.EdgeColor = 'black';
set(gca,'fontsize',11,'fontweigh','bold')
set(gca,'fontsize',11,'fontweigh','bold')
title('\bf Histogram')
saveas(gcf,'ExponentialRV101','epsc')
save('ExponentialRV101','yran')
% End of program
% --------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 평균이 $\mu = 2$인 지수확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 이 경우에 비율모수는 $\lambda = \frac{1}{\mu} = 0.5$이다. 또한, 이 확률분포에서 500개 지수난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.9에 그려져 있다.

지수확률분포에 관한 MATLAB함수들로는 exppdf, pdf, expcdf, cdf, expinv, icdf, expstat, expfit, mle, fitdist, dfittool, explike, exprnd, random, randtool 등이 있다. ■

예제 1.5.10 R을 이용해서 지수확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 지수난수를 살펴보기 위해서, 다음 R프로그램 ExponentialRV101R.R을 실행해 보자.

```
# --------------------------------------------------------------------------
# Filename: ExponentialRV101R.R
# Exponential Random Variable
# Programmed by CBS
# --------------------------------------------------------------------------
```

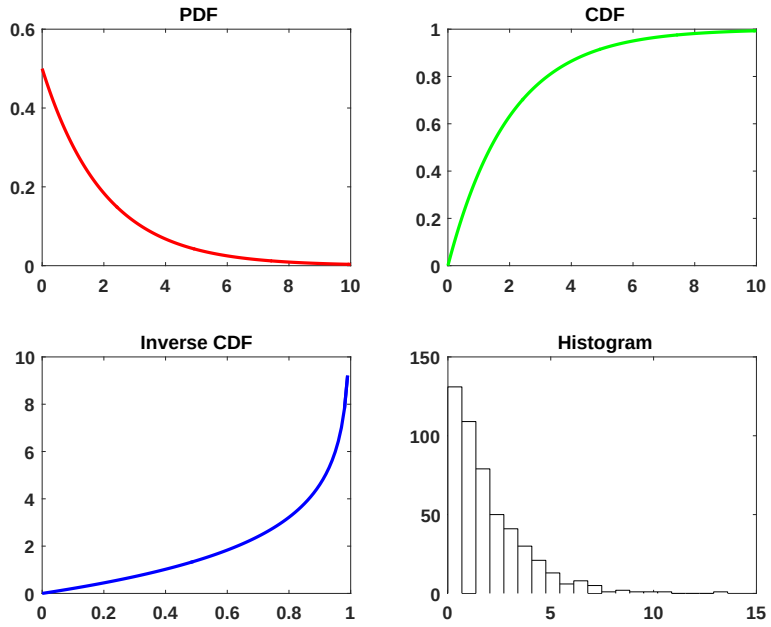


그림 1.5.9. 지수확률변수

```
nAbsissa <- 801
x <- seq(0,10,len=nAbsissa)
mu <- 2; ratee <- 1/mu
yp <- dexp(x, rate=ratee, log=FALSE)   # PDF
yc <- pexp(x, rate=ratee, lower.tail=TRUE, log.p=FALSE)  # CDF
ix <- seq(0,1,len=nAbsissa)
yi <- qexp(ix, rate=ratee, lower.tail=TRUE, log.p=FALSE) #Inverse CDF
nSim <- 500
set.seed(41)
yran <- rexp(nSim, rate=ratee)

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
# postscript('ExponentialRV101R.eps')  # Start to save figure
RVdata1 <- data.frame(x,yp,yc)
RVdata2 <- data.frame(ix,yi)
RVdata3 <- data.frame(yran)
plot11 <- ggplot(RVdata1, aes(x,yp)) +
          geom_line(col="red",lwd=1.2) +
          xlab("x") +
          ggtitle("PDF")
plot12 <- ggplot(RVdata1, aes(x,yc)) +
          geom_line(col="dark green",lwd=1.2) +
          xlab("x") +
          ggtitle("CDF")
plot13 <- ggplot(RVdata2, aes(ix,yi)) +
          geom_line(col="blue",lwd=1.2) +
          xlab("x") +
          ggtitle("Inverse CDF")
plot14 <- ggplot(RVdata3, aes(x=yran)) +
          geom_histogram(bins=20,fill="white",color="black")+
```

```
        xlab("x") +
        ggtitle("Histogram")
pushViewport(viewport(layout=grid.layout(2,2)))
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot13, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot14, vp=viewport(layout.pos.row=2, layout.pos.col=2))
# dev.off()            # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 평균이 $\mu = 2$인 지수확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 이 경우에 비율모수는 $\lambda = 0.5$이다. 또한, 이 확률분포에서 500개 지수난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.10에 그려져 있다. ■

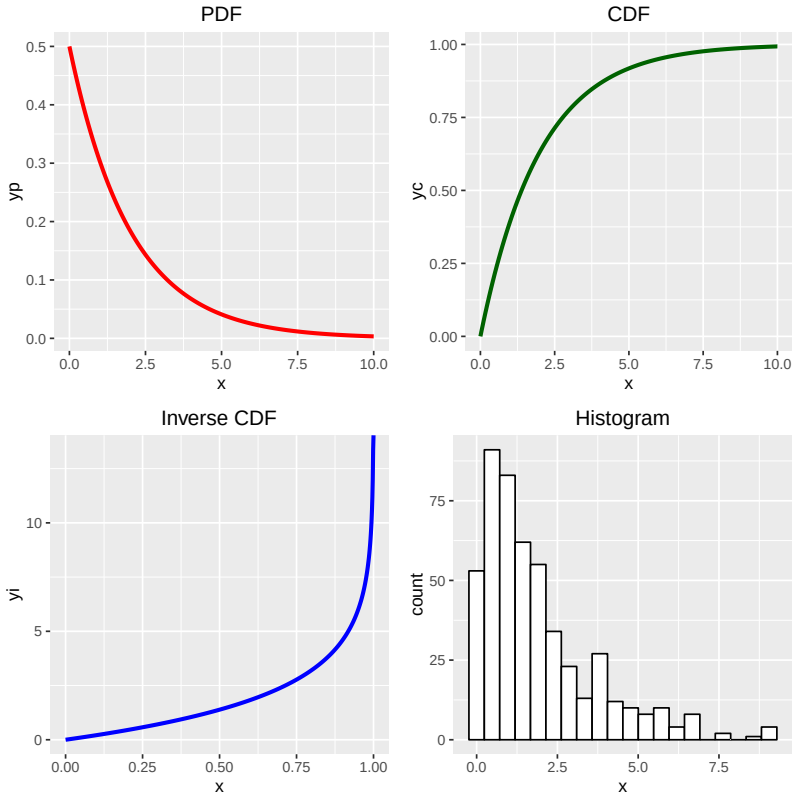


그림 1.5.10. 지수확률변수와 R

1.5.5 Erlang확률분포

서로 독립인 확률변수들 $v_1, v_2, \cdots, v_a$가 평균이 b인 지수확률분포를 따르면, 확률변수 $x \doteq \sum_{i=1}^{a} v_i$는 Erlang확률분포를 따른다고 한다. 식 (1.5.25)에서 알 수 있듯이, 이 Erlang

확률변수의 적률모함수는 다음과 같다.

$$m_x(t) = [1 - bt]^{-a} \qquad \left(t < \frac{1}{b}\right) \tag{1.5.26}$$

여기서 a를 형태모수(shape parameter), b를 척도모수(scale parameter) 그리고 $1/b$를 비율모수(rate parameter)라 부른다. 이 적률모함수에 해당하는 확률밀도함수는 다음과 같다.

$$f(x) = \frac{1}{\Gamma(a)b^a} x^{a-1} \exp\left(-\frac{x}{b}\right) 1_{[0,\infty)}(x) \tag{1.5.27}$$

부분적분을 사용하면, 누적확률분포함수가 다음과 같음을 알 수 있다.

$$F(x) = \left[1 - \exp\left(-\frac{x}{b}\right) \sum_{i=0}^{a-1} \frac{1}{\Gamma(i+1)b^i} x^i\right] 1_{[0,\infty)}(x) \tag{1.5.28}$$

이 Erlang확률분포의 평균과 분산은 각각 다음과 같다.

$$E(x) = b\,a \qquad Var(x) = b^2 a \tag{1.5.29}$$

Erlang확률분포은 대기행렬모형(queueing model)에서 자주 사용된다. 서로 독립이고 지지대가 $(0,1)$인 일양확률변수들 $u_1, u_2, \cdots, u_a$에 대해서, $-b\ln u_1, -b\ln u_2, \cdots, -b\ln u_a$은 척도모수가 b인 지수확률분포를 따른다. 따라서, $-b\ln\prod_{i=1}^{a} u_i$는 모수벡터가 $[a,b]$인 Erlang확률분포를 따른다. 이 성질을 이용해서 Erlang난수들을 생성한다.

예제 1.5.11 Erlang난수들을 생성하기 위해서 다음 MATLAB프로그램 InverseMethodErlang101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: InverseMethodErlang101.m
%  Erlang Random Numbers by Inverse Method
%  Programmed by CBS
%-------------------------------------------------------------------
a = 3, b = 0.5,                  % Parameters
N = 1000                         % No of Random Numbers
rand('twister',5489)
yran = -b*log(prod(rand(a,N),1));
% Goodness-of-Fit Test
binNum = 30; binWidth = 0.3;
edge = (0:binNum)*binWidth
bins = (edge(1:end-1) + edge(2:end))/2
obsNo = hist(yran,bins)          % Observed RN in the bin
if a==1
   cdf = 1 - exp(-edge/b)
else
```

```
    cdf = 1+edge/(a-1)/b;
    for ii=a-2:-1:1
        cdf = 1+edge/b/ii.*cdf;
    end
    cdf = 1 - exp(-edge/b).*cdf    % True CDF
end
expCounts = N*diff(cdf)            % Expected RN in the bin
[H,P_value,stat] = chi2gof(bins,'ctrs',bins,'frequency',obsNo, ...
                           'expected',expCounts)
% Plotting Histogram and PDF
Nheight = obsNo/binWidth/N;
bar(bins,Nheight,1,'w')
set(gca,'fontsize',11,'fontweigh','bold','xlim',[-0.2 10*b])
hold on
xx = linspace(0,binNum-1,501);
pp = gampdf(xx,a,b);               % True PDF
plot(xx,pp,'r','linewidth',2)
legend('Histogram','Erlang PDF',1)
xlabel('\bf x'), ylabel('\bf Relative Frequency')
hold off
saveas(gcf,'InverseMethodErlang101','epsc')
save('InverseMethodErlang101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 형태모수가 $a = 3$이고 척도모수가 $b = 0.5$인 Erlang 확률분포에서 Erlang난수들 1000개를 생성한다. 이 난수들이 Erlang확률분포에서 발생되었는지를 검정하기 위해서 카이제곱검정을 한 결과, 검정통계량값은 8.6761이고 자유도는 14이며 p값은 0.8512이다. 즉, 이 난수들이 Erlang확률분포에서 발생되었다는 귀무가설을 채택한다. 이 Erlang난수들의 히스토그램이 그림 1.5.11에 그려져 있다. ■

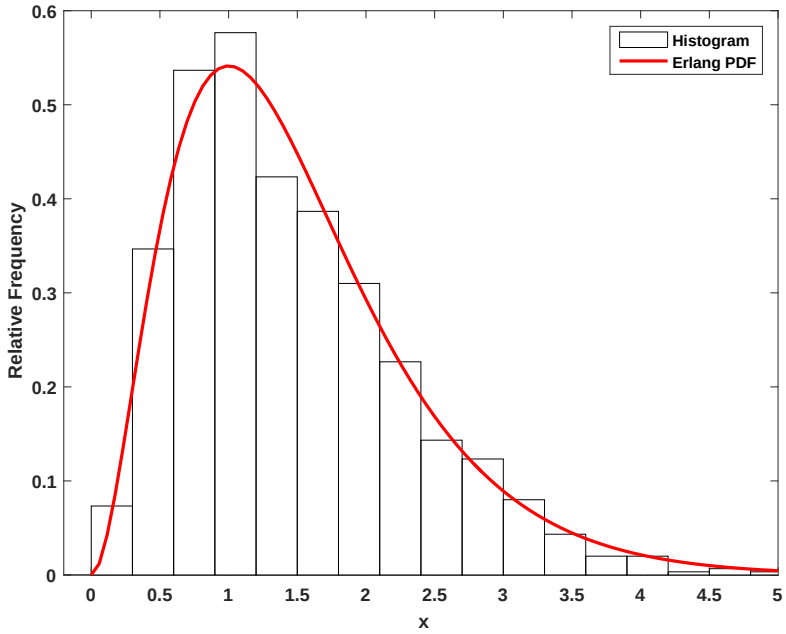


그림 1.5.11. 역함수법과 Erlang확률분포

Erlang확률분포에 관한 MATLAB함수들로는 MATLAB Statistics Toolbox에 들어 있는 stat::erlangPDF, stat::erlangCDF, stat::erlangQuantile, stat::erlangRandom 등이 있다. 다음 소절에서 알 수 있듯이, Erlang확률분포는 감마확률분포의 특수한 경우이다.

1.5.6 감마확률분포

Erlang확률분포에서 형태모수(shape parameter) a를 자연수에서 실수 $\alpha(>0)$로 확장한 것을 감마확률분포라 한다. 확률변수 x가 감마확률분포를 따르는 것을 다음과 같이 표기한다.

$$x \overset{d}{\sim} Gamma(\alpha, \beta) \tag{1.5.30}$$

여기서 α를 형태모수(shape parameter), β를 척도모수(scale parameter), 그리고 $1/\beta$를 비율모수(rate parameter)라 부른다. 이 감마확률분포의 적률모함수는 다음과 같다.

$$m_x(t) = [1-\beta t]^{-\alpha}, \quad \left(t < \frac{1}{\beta}\right) \tag{1.5.31}$$

이 적률모함수에 해당하는 확률밀도함수는 다음과 같다.

$$f_\Gamma(x;\alpha,\beta) = \frac{1}{\Gamma(\alpha)\beta^\alpha} x^{\alpha-1} \exp\left(-\frac{x}{\beta}\right) 1_{[0,\infty)}(x) \tag{1.5.32}$$

또한 누적확률분포함수가 다음과 같다.

$$F_\Gamma(x;\alpha,\beta) = \frac{1}{\Gamma(\alpha)} \gamma\left(\alpha, \frac{x}{\beta}\right) 1_{[0,\infty)}(x) \tag{1.5.33}$$

여기서 불완비감마함수(incomplete gamma function)는 다음과 같다.

$$\gamma(s,x) \doteq \int_0^x t^{s-1} e^{-t} dt \tag{1.5.34}$$

이 감마확률분포의 평균과 분산은 각각 다음과 같다.

$$E(x) = \beta\alpha, \quad Var(x) = \beta^2\alpha \tag{1.5.35}$$

서로 독립인 확률변수들 x_1과 x_2가 각각 $Gamma(\alpha_1,\beta)$과 $Gamma(\alpha_2,\beta)$를 따르는

경우, 적률모함수를 이용해서 다음 식이 성립함을 증명할 수 있다.

$$x_1 + x_2 \overset{d}{\sim} Gamma(\alpha_1 + \alpha_2, \beta) \tag{1.5.36}$$

확률변수 x가 감마확률분포 $Gamma(\alpha, \beta)$를 따르면, 확률변수 $y \doteq cx$는 감마확률분포 $Gamma(\alpha, c\beta)$을 따른다. 따라서, 감마확률분포 $Gamma(\alpha, 1)$에서 생성한 난수들에 β를 곱하면, 감마확률분포 $Gamma(\alpha, \beta)$에서 생성한 난수들을 얻는다.

예제 1.5.12 일반지수확률분포(generalized exponential probability distribution)의 누적확률분포함수는 다음과 같다.

$$F_{GE}(x; \alpha, \beta) = \left[1 - \exp\left(-\frac{x}{\beta}\right)\right]^{\alpha} 1_{[0,\infty)}(x) \tag{1}$$

여기서 α와 β는 양수들이다. 이에 해당하는 확률밀도함수는 다음과 같다.

$$f_{GE}(x; \alpha, \beta) = \frac{\alpha}{\beta} \exp\left(-\frac{x}{\beta}\right) \left[1 - \exp\left(-\frac{x}{\beta}\right)\right]^{\alpha-1} 1_{[0,\infty)}(x) \tag{2}$$

또한, 일반지수확률분포함수의 역함수는 다음과 같다.

$$F_{GE}^{-1}(x; \alpha, \beta) = -\beta \ln\left(1 - x^{1/\alpha}\right) 1_{[0,1)}(x) \tag{3}$$

식 (3)과 역함수법을 적용해서 일반지수난수들을 생성할 수 있다.

각 $x(\geq 0)$에 대해서 다음 식이 성립한다.

$$1 - e^{-x} \leq x \tag{4}$$

식 (4)를 이용해서, 각 $x(\geq 0)$에 대해 다음 식이 성립함을 증명할 수 있다.

$$\frac{1}{\Gamma(\alpha)} x^{\alpha-1} e^{-x} \leq \frac{2^{\alpha}}{\Gamma(\alpha+1)} \frac{\alpha}{2} [1 - e^{-x/2}]^{\alpha-1} e^{-x/2} \tag{5}$$

즉, 다음 식이 성립한다.

$$f_{\Gamma}(x; \alpha, 1) \leq c\, f_{GE}(x; \alpha, 2) \tag{6}$$

여기서 c는 다음과 같다.

$$c \doteq \frac{2^{\alpha}}{\Gamma(\alpha+1)} \tag{7}$$

식 (6)과 채택기각법을 사용해서 일반지수난수들로부터 감마난수들을 생성할 수 있다.

지금까지 설명한 방법으로 감마확률분포 $Gamma(\alpha, 1)$에서 감마난수들을 생성하기 위해서, 다음 MATLAB프로그램 InverseMethodGamma101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: AcceptRejectGamma101.m
%  Generating Gamma RN
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
rng((1001^2+1)/2,'twister')
N = 1000                        % No of Random Numbers
% b = 1; Let Scale parameter = 1
a = 4.5                         % Shape parameter
if a >= 1,
    aFrac = a-floor(a)+1        % Fraction Part 1 <= aFrac < 2
else
    aFrac = a
end
aInte = a - aFrac
% Integer Part
rand('twister',5489)
yranInte = zeros(1,N);
if aInte > 0
    yranInte = -log(prod(rand(aInte,N),1));  % Erlang RN w/ b=1
end
% Fraction Part
N25 = N*2.5;
GEran = - 2*log(1-rand(1,N25).^(1/aFrac)); % General Exponential RN
c = 2^aFrac/gamma(aFrac+1), AccPr = 1/c
ff = 1/gamma(aFrac)*GEran.^(aFrac-1).*exp(-GEran);
gg = aFrac/2*exp(-GEran/2).*(1-exp(-GEran/2)).^(aFrac-1);
yranFrac = GEran(find(rand(1,N25) <= ff./gg/c));  % Gamma RN
yranFrac = yranFrac(1:N);
% Total = Integer + Fraction
yran = yranInte + yranFrac;
% Goodness-of-Fit test
[H,P_value,stat] = chi2gof(yran,'cdf',@(z)gamcdf(z,a,1))
% Plotting Histogram and True PDF
binNo = 20;
[ Nbar,xcenter ] = hist(yran,binNo);          % Histogram
xWidth =  xcenter(2)-xcenter(1)
Nheight = Nbar/xWidth/N;
bar(xcenter,Nheight,1,'w')
set(gca,'fontsize',11,'fontweigh','bold')
hold on
xx = linspace(0,xWidth*(binNo+2),501);
pp = gampdf(xx,a);                             % True PDF
plot(xx,pp,'r','linewidth',2)
xlabel('x'), ylabel('Relative Frequency')
legend('\bf Histogram','\bf Gamma PDF','location','NE')
hold off
saveas(gcf,'AcceptRejectGamma101','epsc')
save('AcceptRejectGamma101','yran')
```

```
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 감마확률분포 $Gamma(\alpha, 1)$에서 감마난수들을 생성한다. 모수 α를 다음과 같이 분해한다.

$$\alpha = \alpha_I + \alpha_F \tag{8}$$

여기서 α_I는 α의 소수부분에 1을 더한 수이다. 우선, α가 1이상인 경우를 살펴보자. 서로 독립인 확률변수들 x_I과 x_F가 각각 $Gamma(\alpha_I, 1)$과 $Gamma(\alpha_F, 1)$를 따른다고 하면, 식 (1.5.36)에서 알 수 있듯이 다음 식이 성립한다.

$$x_I + x_F \overset{d}{\sim} Gamma\,(\alpha_I + \alpha_F, 1) \tag{9}$$

예제 1.5.11에서 설명한 방법을 사용해서, Erlang확률분포 $Gamma\,(\alpha_I, 1)$에서 난수들 $\{y_{I,j}\}$를 생성한다. 또한, 역함수법을 사용해서 일반지수난수들을 생성하고, 이 일반지수난수들과 채택기각법을 사용해서 감마확률분포 $Gamma\,(\alpha_F, 1)$에서 난수들 $\{y_{F,j}\}$을 생성한다. 식 (9)에서 알 수 있듯이, $\{y_j \doteq y_{I,j} + y_{F,j}\}$가 감마확률분포 $Gamma\,(\alpha, 1)$에서 생성된 감마난수들이다. 만약 형태모수 α가 1미만이면, 역함수법을 사용해서 일반지수난수들을 생성하고, 이 일반지수난수들과 채택기각법을 사용해서 감마확률분포 $Gamma\,(\alpha, 1)$에서 난수들 $\{y_j\}$을 생성한다.

이 예제에서는 척도모수 α가 4.5인 감마확률분포 $Gamma(4.5, 1)$에서 감마난수들 1000개를 생성한다. 이 난수들이 감마확률분포에서 발생되었는지를 검정하기 위해서 카이제곱검정을 한 결과, 검정통계량값은 12.0175이고 자유도는 8이며 p값은 0.1504이다. 즉, 이 난수들이 감마확률분포에서 발생되었다는 귀무가설을 채택한다. 이 감마난수들의 히스토그램이 그림 1.5.12에 그려져 있다. 그림 1.5.12에서 이 난수들이 감마확률분포로부터 발생되었다는 것을 알 수 있다. ■

예제 1.5.13 Marsaglia & Tsang [32]은 다음과 같은 채택기각법을 써서 감마확률분포에서 감마난수들을 생성할 것을 제안하였다.

우선 형태모수 α가 1이상인 경우에 다음 알고리즘을 적용한다.

(1단계) 상수 $f = \alpha - 1/3$와 상수 $c = 1/\sqrt{9f}$를 계산한다.

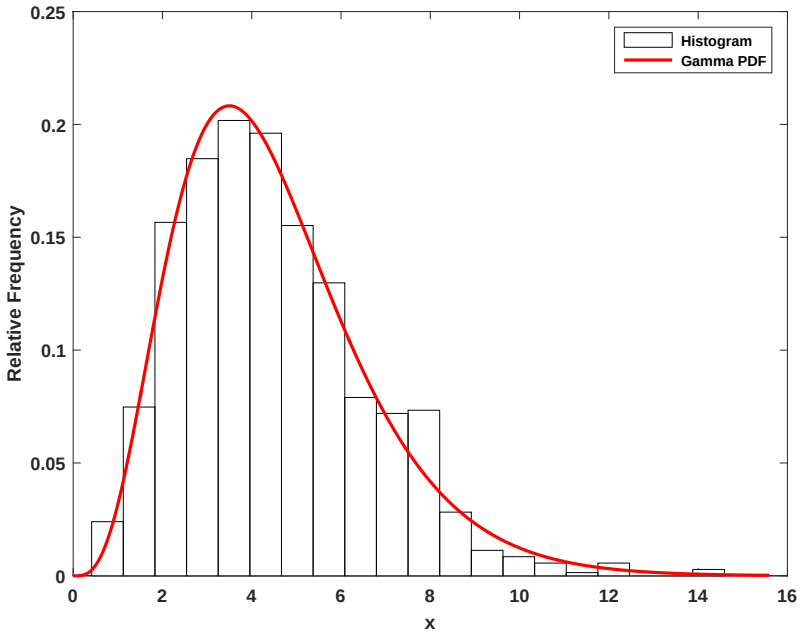


그림 1.5.12. 역함수법과 감마확률분포

(2단계) 서로 독립적인 표준정규난수 z와 지지대가 $(0,1)$인 일양난수를 생성한다. 또한, 상수 $w = [1+cz]^3$를 계산한다.

(3단계) 만약 $z > -1/c$이고 또한 만약 $\ln u < 0.5z^2 + f - fw + f\ln w$ 이면, $x = fw$를 새로운 감마난수로 채택한다. 그렇지 않으면, 제2단계로 돌아간다.

형태모수 α가 1보다 작은 경우에 다음 알고리즘을 적용한다.

(1단계) 상수 $f = 0.07 + 0.75\sqrt{1-\alpha}$와 상수 $b = 1 + e^{-f}\alpha/f$를 계산한다.

(2단계) 서로 독립적이고 지지대가 $(0,1)$인 일양난수들 u_1과 u_2를 생성한다. 또한, 상수 $w = bu_1$를 계산한다.

(3단계) 만약 $w \le 1$이면, $x = fw^{1/\alpha}$라 놓는다. 만약 $w \le 1$이면, $x = fw^{1/\alpha}$라 놓는다.

(4단계) 만약 $u_2 \le [2-x]/[2+x]$이 성립하면, x를 감마난수로 채택한다. 만약 $u_2 > [2-x]/[2+x]$이 성립하고 또한 만약 $u_2 \le e^{-x}$이 성립하면, x를 감마난수로 채택한다. 그렇지 않으면, 제3단계로 돌아간다.

지금까지 설명한 방법으로, 감마확률분포 $Gamma(\alpha,1)$에서 감마난수들을 생성하기 위해서 다음 MATLAB프로그램 RANDGuse101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: RANDGuse101.m
%  Gamma Random Numbers from Gamma(a,1)
%  by Marsaglia & Tsang Method (2000)
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
rng((1001^2+1)/2,'twister')
n = 1000                              % Number of Random Numbers
v1 = randg;
v2 = randg(2.6,2)
v3 = randg(0.8,2,3)
rand('twister',5489)
% Plotting Histogram and True PDF
a = 4.5                               % Parameteter
N = 1000;                             % Number of Random Numbers
binNo = 20;
[ Nbar,xcenter ] = hist(randg(a,1,N),binNo);  % Histogram
xWidth =  xcenter(2)-xcenter(1)
Nheight = Nbar/xWidth/N;
bar(xcenter,Nheight,1,'y')
set(gca,'fontsize',11,'fontweigh','bold')
hold on
xx = linspace(0,xWidth*(binNo+2),501);
pp = gampdf(xx,a);                              % True PDF
plot(xx,pp,'r','linewidth',2)
xlabel('x'), ylabel('Relative Frequency')
legend('\bf Histogram','\bf Gamma PDF','location','NE')
hold off
saveas(gcf,'RANDGuse101','epsc')
save('RANDGuse101','Nbar','xcenter')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 앞에서 기술한 방법으로 감마난수들을 생성한다. MATLAB명령문 '[Nbar, xcenter] = hist(randg(a,1,N),bino)'는 MATLAB함수 randg를 사용해서 생성한 1000개 감마난수들의 히스토그램을 작성한다. 이 감마난수들의 히스토그램이 그림 1.5.13에 그려져 있다. 그림 1.5.13에서 이 난수들이 감마확률분포로부터 발생되었다는 것을 알 수 있다. ■

예제 1.5.14 MATLAB을 이용해서 감마확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 감마난수를 살펴보기 위해서, 다음 MATLAB프로그램 GammaRV101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: GammaRV101.m
%  Gamma Random Variable
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all
alpha = 4.3, beta = 0.5                         % Parameters
```

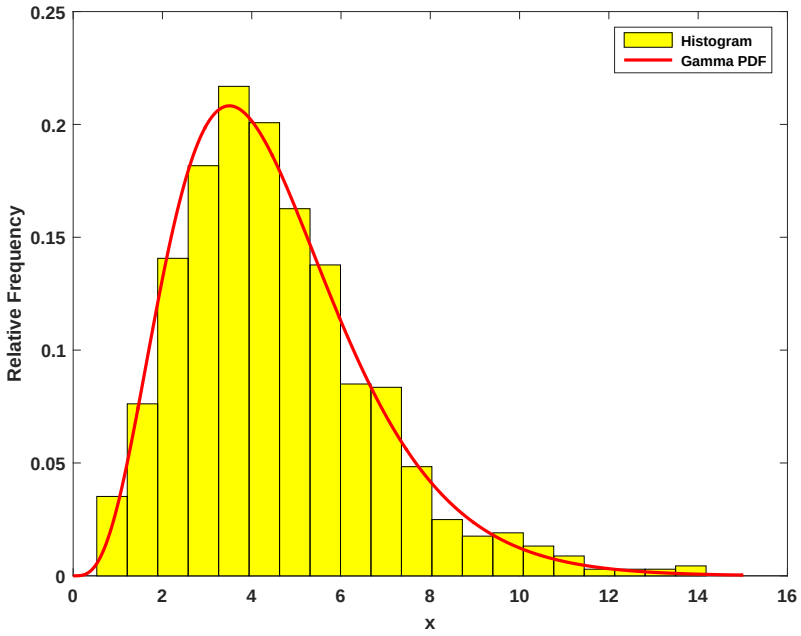


그림 1.5.13. Marsaglia-Tsang법

```
% PDF
yp1 = pdf('gam',1,alpha,beta)
yp2 = gampdf(1,alpha,beta)
xx = 0:0.02:10;
yy = pdf('Gamma',xx,alpha,beta);
subplot(2,2,1)
plot(xx,yy,'r-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 6.1],'ylim',[0 0.5])
title('\bf PDF')
% CDF
yc1 = cdf('gam',1,alpha,beta)
yc2 = gamcdf(1,alpha,beta)
xx = 0:0.02:6.1;
yy = cdf('Gamma',xx,alpha,beta);
subplot(2,2,2)
plot(xx,yy,'g-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 6.1],'ylim',[0 1])
title('\bf CDF')
% Inverse CDF
yi1 = icdf('gam',0.5,alpha,beta)
yi2 = gaminv(0.5,alpha,beta)
xx = 0:0.01:1;
yy = icdf('Gamma',xx,alpha,beta);
subplot(2,2,3)
plot(xx,yy,'b-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[0 6.1])
title('\bfInverse CDF')
% Random Numbers
yr1 = random('gam',alpha,beta,3)
yr1 = gamrnd(alpha,beta,3)
rand('twister',5489)
yran = random('Gamma',alpha,beta,1,500);
subplot(2,2,4)
h1 = histogram(yran)
h1.NumBins = 20;
h1.FaceColor = [1 1 1];
```

```
h1.EdgeColor = 'black';
set(gca,'fontsize',11,'fontweigh','bold')
title('\bf Histogram')
saveas(gcf,'GammaRV101','epsc')
save('GammaRV101','yran')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 형태모수가 $\alpha = 4.3$이고 척도모수(scale parameter)가 $\beta = 0.5$인 감마확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수가 그려진다. 또한, 이 확률분포에서 500개 감마난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림그림 1.5.14에 그려져 있다.

감마확률분포에 관한 MATLAB함수들로는 gampdf, pdf, gamcdf, cdf, gaminv, icdf, gamstat, gamfit, mle, fitdist, dfittool, gamlike, gamrnd, randg, random, randtool 등이 있다. ■

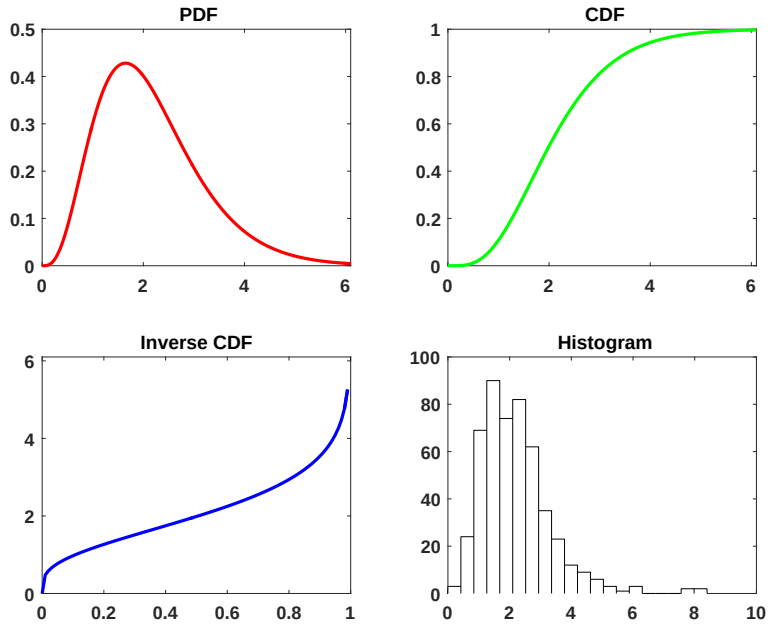


그림 1.5.14. 감마확률변수와 MATLAAB

예제 1.5.15 R을 이용해서 감마확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 감마난수를 살펴보기 위해서, 다음 R프로그램 GammaRV101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: GammaRV101R.R
# Gamma Random Variable
# Programmed by CBS
```

```
# ------------------------------------------------------------------------
nAbsissa <- 801
x <- seq(0,6,len=nAbsissa)
yp <- dgamma(x, shape=4.3, scale=0.5, log=FALSE)   # PDF
yc <- pgamma(x, shape=4.3, scale=0.5, lower.tail=TRUE, log.p=FALSE)  # CDF
ix <- seq(0,1,len=nAbsissa)
yi <- qgamma(ix, shape=4.3, scale=0.5, log.p=FALSE) #Inverse CDF
nSim <- 500
set.seed(41)
yran <- rgamma(nSim, shape=4.3, scale=0.5)

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('GammaRV101R.eps')  # Start to save figure
RVdata1 <- data.frame(x,yp,yc)
RVdata2 <- data.frame(ix,yi)
RVdata3 <- data.frame(yran)
plot11 <- ggplot(RVdata1, aes(x,yp)) +
          geom_line(col="red",lwd=1.2) +
          xlab("x") +
          ggtitle("PDF")
plot12 <- ggplot(RVdata1, aes(x,yc)) +
          geom_line(col="green",lwd=1.2) +
          xlab("x") +
          ggtitle("CDF")
plot13 <- ggplot(RVdata2, aes(ix,yi)) +
          geom_line(col="blue",lwd=1.2) +
          xlab("x") +
          ggtitle("Inverse CDF")
plot14 <- ggplot(RVdata3, aes(x=yran)) +
          geom_histogram(bins=20,fill="white",color="black")+
          xlab("x") +
          ggtitle("Histogram")
pushViewport(viewport(layout=grid.layout(2,2)))
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot13, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot14, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()             # End to save figure
# ------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 형태모수가 $\alpha = 4.3$이고 척도모수(scale parameter)가 $\beta = 0.5$인 감마확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수가 그려진다. 또한, 이 확률분포에서 500개 감마난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.15에 그려져 있다. ■

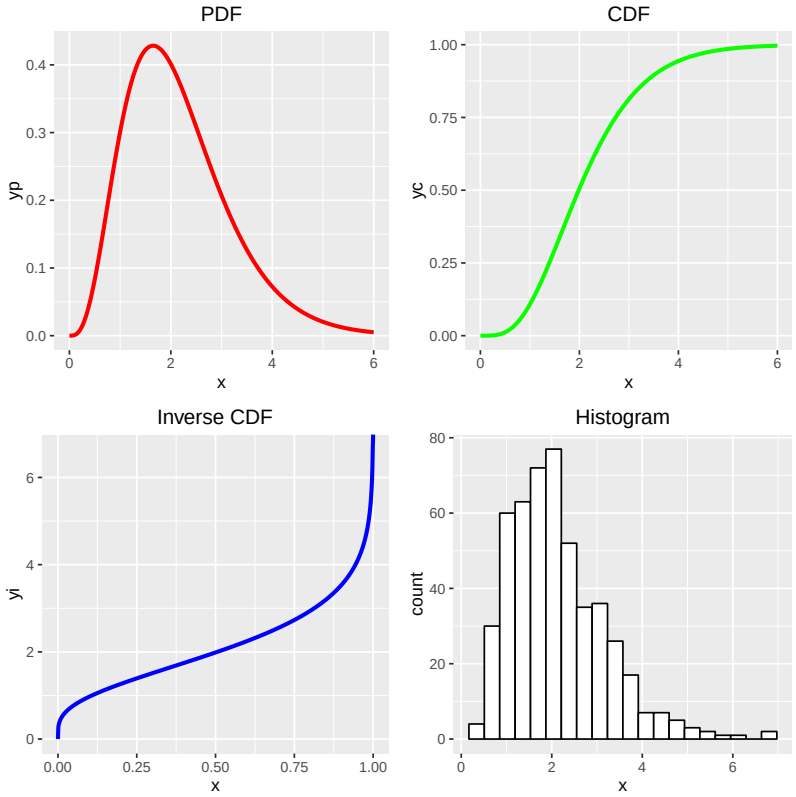


그림 1.5.15. 감마확률변수와 R

1.5.7 카이제곱확률분포

서로 독립인 표준정규확률변수들 $v_1, v_2, \cdots, v_\nu$ 의 제곱합 $x \doteq \sum_{i=1}^{\nu} v_i^2$ 을 자유도가 ν 인 카이제곱확률변수라 부르고, 이에 해당하는 카이제곱확률분포를 χ_v^2 로 표기한다. 자유도가 ν 인 카이제곱확률변수의 확률밀도함수는 다음과 같다.

$$f(x) = \frac{1}{2^{\nu/2}\Gamma(\frac{\nu}{2})} x^{[\nu-2]/2} \exp\left(-\frac{x}{2}\right) 1_{[0,\infty)}(x) \tag{1.5.37}$$

식 (1.5.32) 와 식 (1.5.37) 에서 알 수 있듯이, 자유도가 ν 인 카이제곱확률변수는 척도모수가 2 이고 형태모수가 $\nu/2$ 인 감마확률분포를 따른다. 식 (1.5.35) 에서 알 수 있듯이, 이 카이제곱확률분포의 평균과 분산은 각각 다음과 같다.

$$E(x) = \nu \qquad Var(x) = 2\nu \tag{1.5.38}$$

카이제곱확률분포를 χ_ν^2 에서 난수들을 생성하기 위해서는, 우선 표준정규난수들을 생성한 다음 ν 개 표준정규난수들의 제곱합을 구한다. 그러나, Erlang난수를 생성하는 방법을 변형해서 좀 더 빠르게 카이제곱난수들을 구할 수 있다. 만약 ν 가 짝수이면, 서로 독립이고 지지대가

$(0,1)$ 인 일양확률변수들 $u_1, u_2, \cdots, u_m$ 에 대해서, $-\frac{1}{2}u_1, -\frac{1}{2}u_2, \cdots, -\frac{1}{2}u_{\nu/2}$ 은 척도모수가 $\frac{1}{2}$ 인 지수확률분포를 따른다. 따라서, $-\frac{1}{2}\ln\prod_{i=1}^{\nu/2} u_i$ 는 척도모수가 2 이고 형태모수가 $\nu/2$ 인 감마확률분포 $Gamma(\nu/2, 2)$ 를 따른다. 만약 ν 가 홀수이면, 자유도가 $\nu-1$ 인 카이제곱확률변수와 표준정규확률변수의 제곱을 더하면 자유도가 ν 인 카이제곱확률변수라는 성질을 이용한다.

예제 1.5.16 카이제곱난수들을 생성하기 위해서 다음 MATLAB프로그램 InverseMethod-ChiSquare101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: InverseMethodChiSquare101.m
%  Chi-Squared Random Numbers by Inverse Method
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
nu = 9                          % Parameters
N = 100                         % No of Random Numbers
rand('twister',5489)
if mod(nu,2)==0
   yran = -2*log(prod(rand(nu/2,N),1));
else
   yran = -2*log(prod(rand((nu-1)/2,N),1)) + randn(1,N).^2;
end
yran = yran';
% Kolmogorov-Smirnov Test
CDF = chi2cdf(yran,nu);
[H,P_value,KSstat,CV] = kstest(yran,[yran,CDF],0.05)
% Plotting Histogram and PDF
binNum = 30; binWidth = 1;
edge = (0:binNum)*binWidth;
bins = (edge(1:end-1) + edge(2:end))/2;
Nheight = hist(yran,bins)/binWidth/N;
bar(bins,Nheight,1,'w')
set(gca,'fontsize',11,'fontweigh','bold')
hold on
xx = linspace(0,binNum-1,501);
pp = chi2pdf(xx,nu);            % True PDF
plot(xx,pp,'r','linewidth',2)
legend('Histogram','Chi-Square PDF',1)
xlabel('\bf x'), ylabel('\bf Relative Frequency')
axis([-0.2 3*nu+0.2 0 max(Nheight*1.1) ])
hold off
saveas(gcf,'InverseMethodChiSquare101','epsc')
save('InverseMethodChiSquare101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 자유도가 $\nu = 9$ 인 카이제곱난수들 100 개를 생성한다. 이 난수들이 카이제곱확률분포에서 발생되었는지를 검정하기 위해서 Kolmogorov-Smirnov 검정을 한 결과, 검정통계량값은 0.0792 이고 임계값 (critical value) 은 0.1340 이며 p 값은 0.5316 이다. 따라서, 이 난수들이 자유도가 $\nu = 9$ 인 카이제곱확률분포에서 발생되었다는

귀무가설을 채택한다. 이 카이제곱난수들의 히스토그램이 그림 1.5.16에 그려져 있다. ■

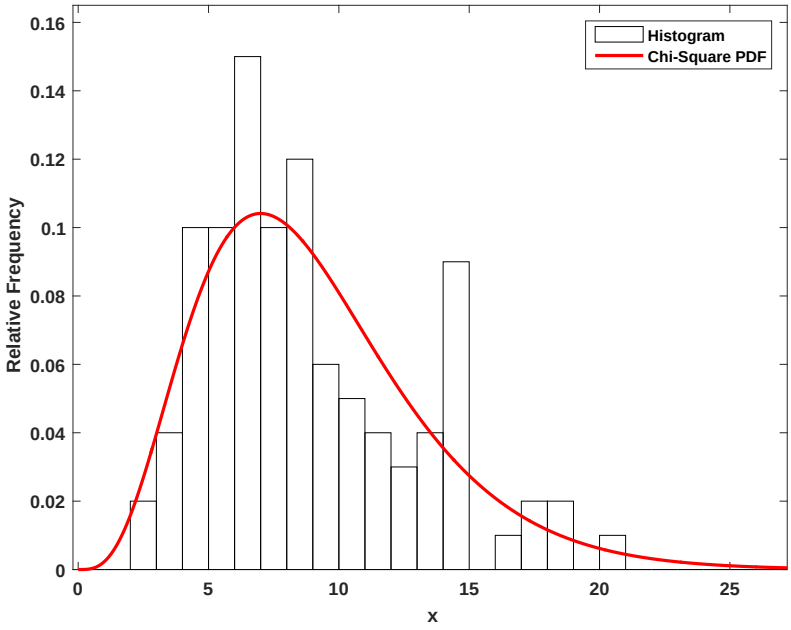


그림 1.5.16. 역함수법과 카이제곱확률분포

예제 1.5.17 MATLAB을 이용해서 카이제곱확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 카이제곱난수를 살펴보기 위해서, 다음 MATLAB프로그램 ChiSquareRV101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: ChiSquareRV101.m
%  Chi-Squared Random Variable
%  Programmed by CBS
%-------------------------------------------------------------------------
clear all, close all
nu = 9                                  % Parameter
% PDF
yp1 = pdf('chi2',1,nu)
yp2 = chi2pdf(1,nu)
xx = 0:0.05:25;
yy = pdf('Chisquare',xx,nu);
subplot(2,2,1)
plot(xx,yy,'r-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 25],'ylim',[0 0.12])
title('\bf PDF')
% CDF
yc1 = cdf('chi2',1,nu)
yc2 = chi2cdf(1,nu)
xx = 0:0.05:25;
yy = cdf('Chisquare',xx,nu);
subplot(2,2,2)
plot(xx,yy,'g-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 25],'ylim',[0 1])
title('\bf CDF')
% Inverse CDF
```

```
yi1 = icdf('chi2',0.5,nu)
yi2 = chi2inv(0.5,nu)
xx = 0:0.01:1;
yy = icdf('Chisquare',xx,nu);
subplot(2,2,3)
plot(xx,yy,'b-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[0 25])
title('\bfInverse CDF')
% Random Numbers
yr1 = random('chi2',nu,3)
yr1 = chi2rnd(nu,3)
rand('twister',5489)
yran = random('Chisquare',nu,1,500);
subplot(2,2,4)
h1 = histogram(yran)
h1.NumBins = 20;
h1.FaceColor = [1 1 1];
h1.EdgeColor = 'black';
set(gca,'fontsize',11,'fontweigh','bold')
title('\bf Histogram')
saveas(gcf,'ChiSquareRV101','epsc')
save('ChiSquareRV101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 자유도가 $\nu = 9$인 카이제곱확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 500개 카이제곱난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.17에 그려져 있다.

카이제곱확률분포에 관한 MATLAB함수들로는 chi2pdf, pdf, chi2cdf, cdf, chi2inv, icdf, chi2stat, chi2fit, chi2rnd, random, randtool 등이 있다. ■

예제 1.5.18 R을 이용해서 카이제곱확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 카이제곱난수를 살펴보기 위해서, 다음 R프로그램 ChiSquareRV101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: ChiSquareRV101R.R
# Chi-squared Random Variable
# Programmed by CBS
# ------------------------------------------------------------------------
nAbsissa <- 801
nu <- 9
x <- seq(0,25,len=nAbsissa)
yp <- dchisq(x, df=nu, ncp=0, log=FALSE)
yc <- pchisq(x, df=nu, ncp=0, lower.tail=TRUE, log.p=FALSE)
ix <- seq(0,1,len=nAbsissa)
yi <- qchisq(ix, df=nu, ncp=0, lower.tail=TRUE, log.p=FALSE)
nSim <- 500
set.seed(41)
yran <- rchisq(nSim, df=nu, ncp=0)
```

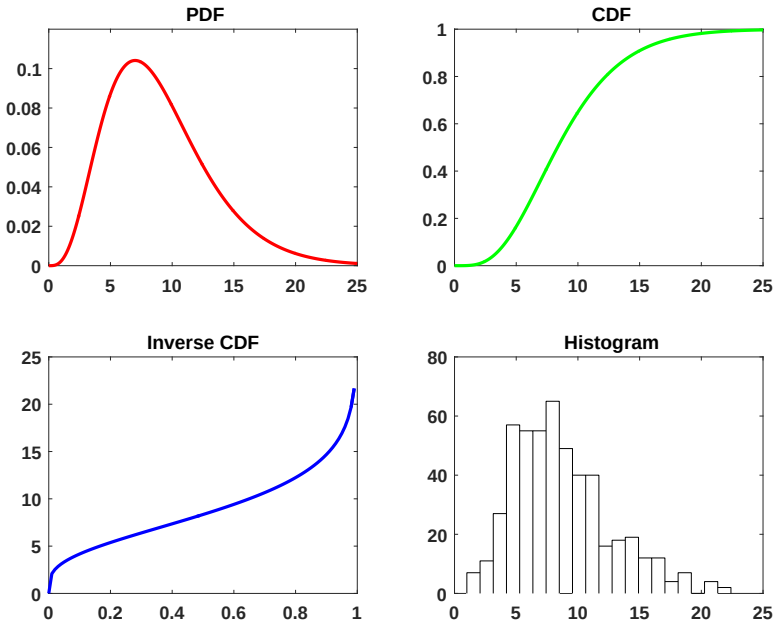


그림 1.5.17. 카이제곱확률변수

```
16
17 # Plotting
18 # install.packages("ggplot2")
19 library(ggplot2)
20 # install.packages("grid")
21 library(grid)
22 setEPS()
23 plot.new()
24 postscript('ChiSquareRV101R.eps')  # Start to save figure
25 RVdata1 <- data.frame(x,yp,yc)
26 RVdata2 <- data.frame(ix,yi)
27 RVdata3 <- data.frame(yran)
28 plot11 <- ggplot(RVdata1, aes(x,yp)) +
29          geom_line(col="red",lwd=1.2) +
30          xlab("x") +
31          ggtitle("PDF")
32 plot12 <- ggplot(RVdata1, aes(x,yc)) +
33          geom_line(col="green",lwd=1.2) +
34          xlab("x") +
35          ggtitle("CDF")
36 plot13 <- ggplot(RVdata2, aes(ix,yi)) +
37          geom_line(col="blue",lwd=1.2) +
38          xlab("x") +
39          ggtitle("Inverse CDF")
40 plot14 <- ggplot(RVdata3, aes(x=yran)) +
41          geom_histogram(bins=20,fill="white",color="black")+
42          xlab("x") +
43          ggtitle("Histogram")
44 pushViewport(viewport(layout=grid.layout(2,2)))
45 print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
46 print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
47 print(plot13, vp=viewport(layout.pos.row=2, layout.pos.col=1))
48 print(plot14, vp=viewport(layout.pos.row=2, layout.pos.col=2))
49 dev.off()             # End to save figure
50 # ------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 자유도가 $\nu = 9$인 카이제곱확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수가 그려진다. 또한, 이 확률분포에서 500개 카이제곱 난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.18에 그려져 있다.

■

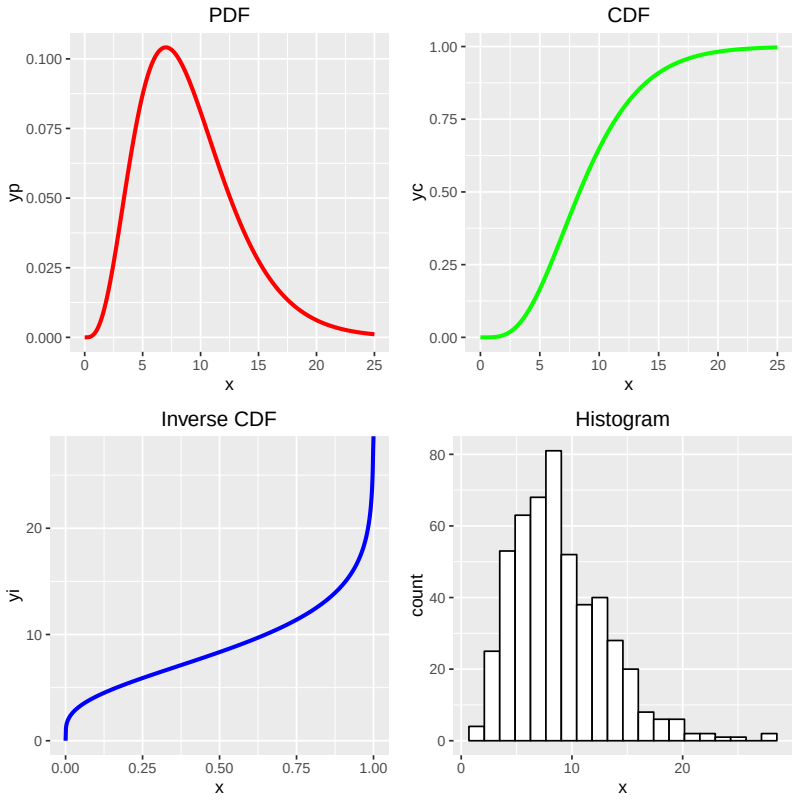


그림 1.5.18. 카이제곱확률변수와 R

1.5.8 t확률분포

서로 독립인 표준정규확률변수 z와 자유도가 ν인 카이제곱확률변수 y에 대해서 다음과 같은 확률변수를 정의하자.

$$x \doteq \frac{z}{\sqrt{y/\nu}} \tag{1.5.39}$$

이 확률변수 x를 자유도가 ν인 t확률변수라고 부르고, 이에 해당하는 확률분포를 t_ν로 표기한다. 이 확률변수의 확률밀도함수는 다음과 같다.

$$f(x) = \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\pi\nu}\Gamma(\frac{\nu}{2})}\left[1 + \frac{x^2}{\nu}\right]^{-[\nu+1]/2} \tag{1.5.40}$$

또한, 평균과 분산은 각각 다음과 같다.

$$E(x) = 0,\ (\nu > 1), \qquad Var(x) = \frac{\nu}{\nu - 2},\ (\nu > 2) \tag{1.5.41}$$

정의에서 알 수 있듯이, 표준정규난수와 카이제곱난수를 이용해서, 확률분포 t_ν로부터 난수를 생성할 수 있다.

예제 1.5.19 MATLAB을 이용해서 t확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 t난수를 살펴보기 위해서, 다음 MATLAB프로그램 tRV101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: tRV101.m
%  t Random Variable
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
nu = 11                                  % Parameter
% PDF
yp1 = pdf('t',1,nu)
yp2 = tpdf(1,nu)
xx = -4:0.01:4;
yy = pdf('T',xx,nu);
subplot(2,2,1)
plot(xx,yy,'r-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold', ...
    'xlim',[-4 4],'ylim',[0 0.5])
title('\bf PDF')
% CDF
yc1 = cdf('t',1,nu)
yc2 = tcdf(1,nu)
xx = -4:0.01:4;
yy = cdf('T',xx,nu);
subplot(2,2,2)
plot(xx,yy,'g-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[-4 4],'ylim',[0 1])
title('\bf CDF')
% Inverse CDF
yi1 = icdf('t',10,nu)
yi2 = tinv(10,nu)
xx = 0:0.01:1;
yy = icdf('T',xx,nu);
subplot(2,2,3)
plot(xx,yy,'b-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[-4 4])
title('\bfInverse CDF')
% Random Numbers
yr1 = random('t',nu,3)
yr1 = trnd(nu,3)
rand('twister',5489)
yran = random('T',nu,1,500);
subplot(2,2,4)
h1 = histogram(yran)
h1.NumBins = 20;
```

```
h1.FaceColor = [1 1 1];
h1.EdgeColor = 'black';
set(gca,'fontsize',11,'fontweigh','bold')
title('\bf Histogram')
saveas(gcf,'tRV101','epsc')
save('tRV101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 자유도가 $\nu = 11$인 t확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 500개 t난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.19에 그려져 있다.

이 확률분포에 관한 MATLAB함수들로는 tpdf, pdf, tcdf, cdf, tinv, icdf, tstat, trnd, random, randtool 등이 있다. ■

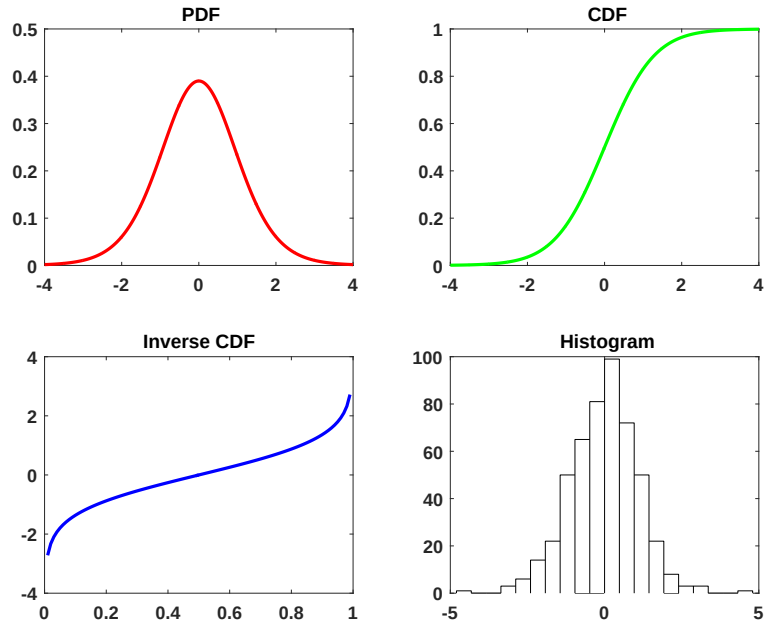


그림 1.5.19. t확률변수

예제 1.5.20 R을 이용해서 t확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 t난수를 살펴보기 위해서, 다음 R프로그램 tRV101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: tRV101R.R
# t Random Variable
# Programmed by CBS
# ------------------------------------------------------------------------
nAbsissa <- 801
nu <- 11
x <- seq(-4,4,len=nAbsissa)
yp <- dt(x, df=nu, ncp=0, log=FALSE)
```

```
yc <- pt(x, df=nu, ncp=0, lower.tail=TRUE, log.p=FALSE)
ix <- seq(0,1,len=nAbsissa)
yi <- qt(ix, df=nu, ncp=0, lower.tail=TRUE, log.p=FALSE)
nSim <- 500
set.seed(41)
yran <- rt(nSim, df=nu, ncp=0)

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('tRV101R.eps')  # Start to save figure
RVdata1 <- data.frame(x,yp,yc)
RVdata2 <- data.frame(ix,yi)
RVdata3 <- data.frame(yran)
plot11 <- ggplot(RVdata1, aes(x,yp)) +
          geom_line(col="red",lwd=1.2) +
          xlab("x") +
          ggtitle("PDF")
plot12 <- ggplot(RVdata1, aes(x,yc)) +
          geom_line(col="green",lwd=1.2) +
          xlab("x") +
          ggtitle("CDF")
plot13 <- ggplot(RVdata2, aes(ix,yi)) +
          geom_line(col="blue",lwd=1.2) +
          xlab("x") +
          ggtitle("Inverse CDF")
plot14 <- ggplot(RVdata3, aes(x=yran)) +
          geom_histogram(bins=20,fill="white",color="black")+
          xlab("x") +
          ggtitle("Histogram")
pushViewport(viewport(layout=grid.layout(2,2)))
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot13, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot14, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()            # End to save figure
# ------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 자유도가 $\nu = 11$인 t확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수가 그려진다. 또한, 이 확률분포에서 500개 t난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.20에 그려져 있다. ■

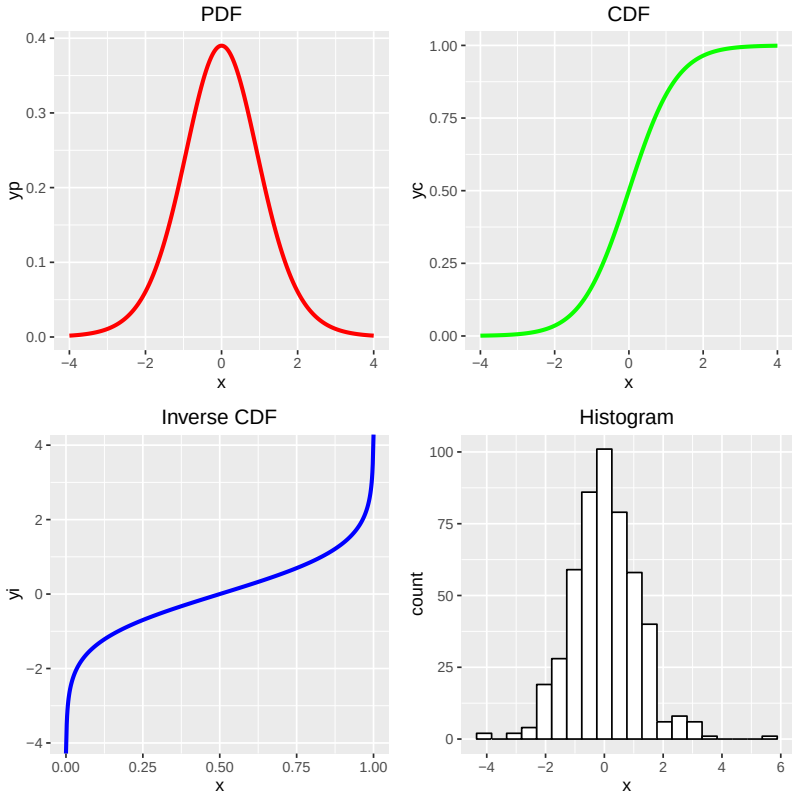


그림 1.5.20. t확률변수와 R

1.5.9 F 확률분포

서로 독립인 자유도가 ν_1 인 카이제곱확률변수 u와 자유도가 ν_2 인 카이제곱확률변수 v에 대해서 다음과 같은 확률변수를 정의하자.

$$x \doteq \frac{u/\nu_1}{v/\nu_2} \tag{1.5.42}$$

이 확률변수 x를 자유도가 (ν_1, ν_2) 인 F 확률변수라고 부르고, 이에 해당하는 확률분포를 F_{ν_1,ν_2} 로 표기한다. 이 확률변수의 확률밀도함수는 다음과 같다.

$$f(x) = \frac{[\nu_1/\nu_2]^{\nu_1/2}}{B(\frac{\nu_1}{2}, \frac{\nu_2}{2})} x^{[\nu_1-2]/2} \left[1 + \frac{\nu_1}{\nu_2} x\right]^{-[\nu_1+\nu_2]/2} \tag{1.5.43}$$

또한, 평균과 분산은 각각 다음과 같다.

$$E(x) = \frac{\nu_2}{\nu_2 - 2}, \ (\nu_2 > 2), \qquad Var(x) = \frac{2\nu_2^2[\nu_1 + \nu_2 - 2]}{\nu_1[\nu_2 - 2]^2[\nu_2 - 4]}, \ (\nu_2 > 4) \tag{1.5.44}$$

정의에서 알 수 있듯이, 서로 독립인 카이제곱난수들을 이용해서 확률분포 F_{ν_1,ν_2} 으로부터

난수를 생성할 수 있다.

예제 1.5.21 MATLAB을 이용해서 F 확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 난수를 살펴보기 위해서, 다음 MATLAB프로그램 F_RV101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: F_RV101.m
%  F Random Variable
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
nu1 = 7, nu2 = 4                    % Parameters
% PDF
yp1 = pdf('f',1,nu1,nu2)
yp2 = fpdf(1,nu1,nu2)
xx = 0:0.02:10;
yy = pdf('F',xx,nu1,nu2);
subplot(2,2,1)
plot(xx,yy,'r-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 9.1],'ylim',[0 0.7])
title('\bf PDF')
% CDF
yc1 = cdf('f',1,nu1,nu2)
yc2 = fcdf(1,nu1,nu2)
xx = 0:0.02:9.1;
yy = cdf('F',xx,nu1,nu2);
subplot(2,2,2)
plot(xx,yy,'g-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 9.1],'ylim',[0 1])
title('\bf CDF')
% Inverse CDF
yi1 = icdf('f',0.5,nu1,nu2)
yi2 = finv(0.5,nu1,nu2)
xx = 0:0.01:1;
yy = icdf('F',xx,nu1,nu2);
subplot(2,2,3)
plot(xx,yy,'b-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[0 9.1])
title('\bfInverse CDF')
% Random Numbers
yr1 = random('f',nu1,nu2,3)
yr1 = frnd(nu1,nu2,3)
rand('twister',5489)
yran = random('F',nu1,nu2,1,100);
subplot(2,2,4)
h1 = histogram(yran)
h1.NumBins = 20;
h1.FaceColor = [1 1 1];
h1.EdgeColor = 'black';
set(gca,'fontsize',11,'fontweigh','bold')
title('\bf Histogram')
saveas(gcf,'F_RV101','epsc')
save('F_RV101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 자유도가 $(7,4)$인 F확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 500개 카이제곱 난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.21에 그려져 있다.

이 F확률분포에 관한 MATLAB함수들로는 fpdf, pdf, fcdf, cdf, finv, icdf, fstat, frnd, random, randtool 등이 있다. ■

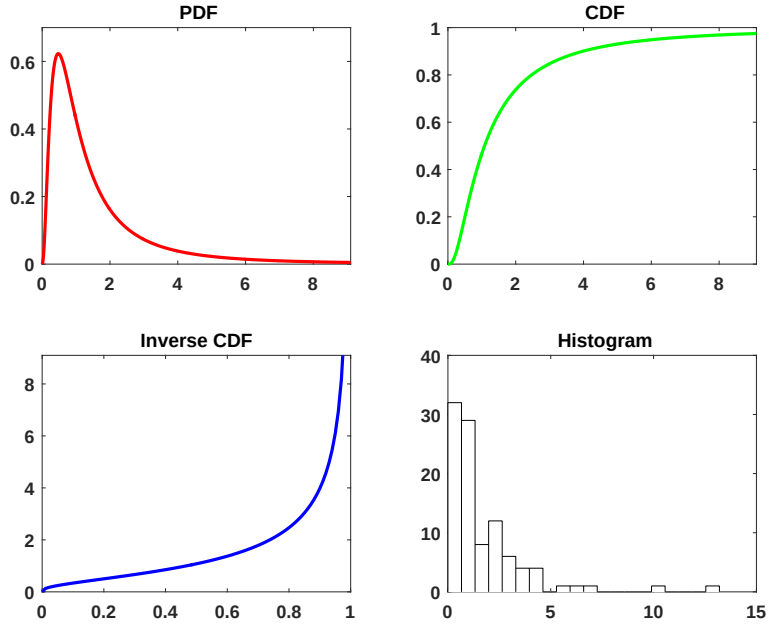


그림 1.5.21. F확률변수와 MATLAB

예제 1.5.22 R을 이용해서 F확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 F난수를 살펴보기 위해서, 다음 R프로그램 F_RV101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------------
# Filename: F_RV101R.R
# F Random Variable
# Programmed by CBS
# ------------------------------------------------------------------------------
nAbsissa <- 801
nu1 <- 7; nu2 <- 4
x <- seq(0,10,len=nAbsissa)
yp <- df(x, df1=nu1, df2=nu2, ncp=0, log=FALSE)
yc <- pf(x, df1=nu1, df2=nu2, ncp=0, lower.tail=TRUE, log.p=FALSE)
ix <- seq(0,1,len=nAbsissa)
yi <- qf(ix, df1=nu1, df2=nu2, ncp=0, lower.tail=TRUE, log.p=FALSE)
nSim <- 500
set.seed(41)
yran <- rf(nSim, df1=nu1,df2=nu2, ncp=0)

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
```

```
library(grid)
setEPS()
plot.new()
postscript('F_RV101R.eps')  # Start to save figure
RVdata1 <- data.frame(x,yp,yc)
RVdata2 <- data.frame(ix,yi)
RVdata3 <- data.frame(yran)
plot11 <- ggplot(RVdata1, aes(x,yp)) +
          geom_line(col="red",lwd=1.2) +
          xlab("x") +
          ggtitle("PDF")
plot12 <- ggplot(RVdata1, aes(x,yc)) +
          geom_line(col="green",lwd=1.2) +
          xlab("x") +
          ggtitle("CDF")
plot13 <- ggplot(RVdata2, aes(ix,yi)) +
          geom_line(col="blue",lwd=1.2) +
          xlab("x") +
          ggtitle("Inverse CDF")
plot14 <- ggplot(RVdata3, aes(x=yran)) +
          geom_histogram(bins=20,fill="white",color="black")+
          xlab("x") +
          ggtitle("Histogram")
pushViewport(viewport(layout=grid.layout(2,2)))
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot13, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot14, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()            # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 자유도가 $(7,4)$인 F확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 500개 F난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.22에 그려져 있다. ■

1.5.10 베타확률분포

형태모수들이 $a(>0)$와 $b(>0)$인 베타확률분포 $Beta(a,b)$의 확률밀도함수는 다음과 같다.

$$f(x) = \frac{1}{B(a,b)} x^{a-1}[1-x]^{b-1} 1_{[0,1]}(x) \tag{1.5.45}$$

이 베타확률분포의 평균과 분산은 각각 다음과 같다.

$$E(x) = \frac{a}{a+b}, \quad Var(x) = \frac{ab}{[a+b]^2[a+b+1]} \tag{1.5.46}$$

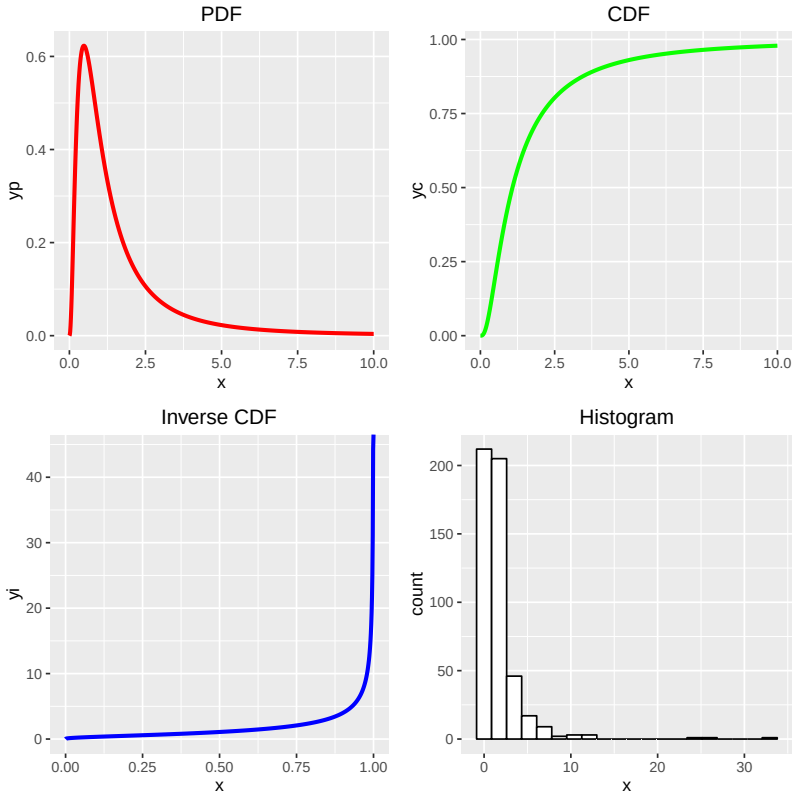


그림 1.5.22. F 확률변수와 R

예제 1.5.23 서로 독립인 확률변수들 x_1과 x_2가 각각 $Gamma(a_1, 1)$과 $Gamma(a_2, 1)$를 따른다고 하면, 이들의 결합확률밀도함수는 다음과 같다.

$$f(x_1, x_2) = \frac{1}{\Gamma(a_1)\Gamma(a_2)} x_1^{a_1-1} x_2^{a_2-1} 1_{[0,\infty]}(x_1) 1_{[0.\infty)}(x_2) \tag{1}$$

다음과 같은 변수변환을 하기로 하자.

$$u = x_1 + x_2 \quad v = \frac{x_1}{x_1 + x_2} \tag{2}$$

다음 식들이 성립한다.

$$x_1 = uv \quad x_2 = u[1 - v] \tag{3}$$

Jacobian은 다음과 같다.

$$J = \left| \frac{\partial(x_1, x_2)}{\partial(u, v)} \right| = \begin{vmatrix} 0 & 1 \\ -1 & 0 \end{vmatrix} = 1 \tag{4}$$

따라서, 확률변수들 u와 v의 결합확률밀도함수는 다음과 같다.

$$\begin{aligned} g(u,v) &= f(x_1(u,v), x_2(u,v))|J| \\ &= \frac{1}{\Gamma(a_1+a_2)} u^{a_1+a_2-1} e^{-u} 1_{[0,\infty)}(u) \frac{\Gamma(a_1+a_2)}{\Gamma(a_1)\Gamma(a_2)} v^{a_1-1}[1-v]^{a_2-1} 1_{[0,1]}(v) \end{aligned} \tag{5}$$

따라서, 확률변수 v의 확률밀도함수는 다음과 같다.

$$f(v) = \frac{1}{B(a_1,a_2)} v^{a_1-1}[1-v]^{a_2-1} 1_{[0,1]}(v) \tag{6}$$

즉, 다음 식이 성립한다.

$$\frac{x_1}{x_1+x_2} \overset{d}{\sim} Beta(a_1,a_2) \tag{7}$$

식 (7)을 이용해서 감마난수들로부터 베타난수들을 생성하기 위해서, 다음 MATLAB파일 BetaRandomNumber101.m을 실행하라.

```
% ------------------------------------------------------------------
%  Filename: BetaRandomNumber101.m
%  Beta Random Numbers from Gamma Random Numbers
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
a = 3, b=5                              % Parameters
N = 100                                 % No of Random Numbers
rand('twister',5489)
x1 = randg(a,N,1); x2 = randg(b,N,1);   % Gamma RN
yran = x1./(x1+x2);                     % Beta RN
% Kolmogorov-Smirnov Test
CDF = betacdf(yran,a,b);
[H,P_value,KSstat,CV] = kstest(yran,[yran,CDF],0.05)
% Plotting Histogram and PDF
binNum = 20; binWidth = 1/binNum;
edge = (0:binNum)*binWidth;
bins = (edge(1:end-1) + edge(2:end))/2;
Nheight = hist(yran,bins)/binWidth/N;
bar(bins,Nheight,1,'w')
set(gca,'fontsize',11,'fontweigh','bold','xlim',[-0.1 1.1])
hold on
xx = linspace(0,binNum-1,501);
pp = betapdf(xx,a,b);           % True PDF
plot(xx,pp,'r','linewidth',2)
legend('Histogram','True PDF',1)
xlabel('\bf x'), ylabel('\bf Relative Frequency')
hold off
saveas(gcf,'BetaRandomNumber101','epsc')
save('BetaRandomNumber101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 베타확률분포 $Beta(3,5)$에서 난수들 100개를 발생한다. MATLAB함수 kstest를 사용해서 Kolmogorov-Smirnov검정을 한 결과, 검정통계량값은 0.0899이고 임계값은 0.1340이며 p값이 0.3720이다. 따라서 이 난수들이 베타확률분포 $Beta(3,5)$에서 생성되었다는 귀무가설이 채택된다. 이 MATLAB프로그램을 실행하면, 그림 1.5.23가 출력된다. 그림 1.5.23에서 막대그래프는 이 난수들의 히스토그램이고, 적색 실선은 진짜확률밀도함수이다. 그림 1.5.23에서 이 난수들이 베타확률분포로부터 발생되었다는 것을 짐작할 수 있다. ■

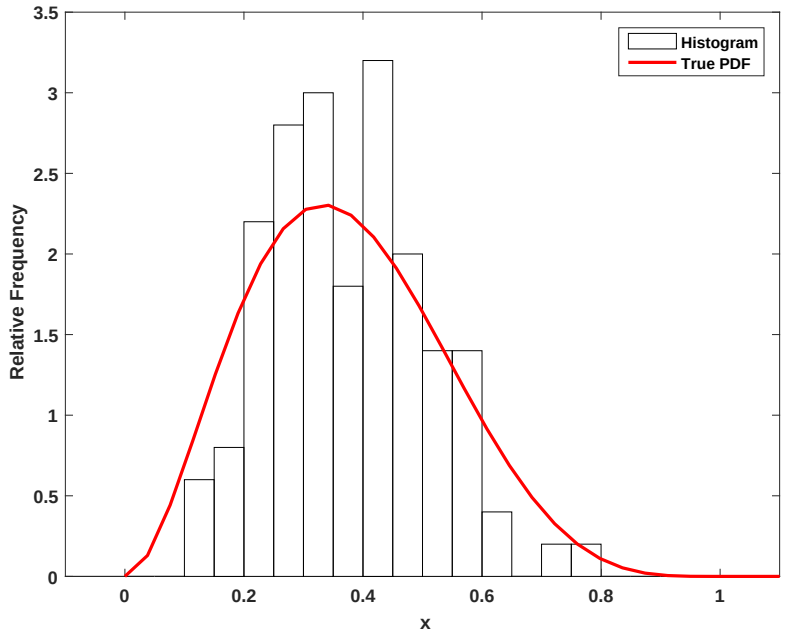


그림 1.5.23. 감마확률변수와 베타확률변수

예제 1.5.24 Jöhnk [20]는 모수들 a와 b가 모두 1 보다 작은 경우 베타확률분포 $Beta(a,b)$에서 베타난수들을 발생시키기 위해서 다음과 같은 채택기각법을 사용할 것을 제안하였다.

(1단계) 지지대가 $(0,1)$인 일양난수들 u_1과 u_2는 생성하고, $x_1 = u_1^{1/a}$와 $x_2 = u_2^{1/b}$를 계산한다.

(2단계) 만약 식 $x_1 + x_2 > 1$이 성립하면, 제1단계로 돌아간다. 그렇지 않으면, $y = x_1/[x_1 + x_2]$를 베타난수로 채택한다.

지금까지 설명한 방법으로, 베타확률분포 $Beta(a,b)$에서 베타난수들을 생성하기 위해서 다음 MATLAB프로그램 AcceptRejectBeta102.m을 실행해 보자.

```
% ----------------------------------------------------------------
```

```
%  Filename: AcceptRejectBeta102.m
%  Johnk, M. D. (1964), Metrika, vol. 8, pp. 5-15.
%  Generating Beta Random Numbers by Acceptance-Rejection Method
%  Programmed by CBS
%-------------------------------------------------------------------
clear all, close all
a = 0.3, b = 0.6                    % Parameters
N = 1000                            % No of Random Numbers
N25 = N*2.5
rand('twister',5489)
u1 = rand(N25,1); u2 = rand(N25,1);
x1 = u1.^(1/a); x2 = u2.^(1/b);
inde = find(x1+x2 <= 1);
x1 = x1(inde); x1 = x1(1:N);
x2 = x2(inde); x2 = x2(1:N);
yran = x1./(x1+x2);
% Goodness-of-Fit Test
[H,P_value,stat] = chi2gof(yran,'cdf',{@betacdf,a,b})
% Plotting Histogram and PDF
Nheight = hist(yran,0.025:0.05:0.975)/0.05/N;
bar(0.025:0.05:0.975,Nheight,1,'w')
set(gca,'fontsize',11,'fontweigh','bold','xlim',[-0.025 1.025 ])
hold on
xx = linspace(0,1,501);
pp = betapdf(xx,a,b);               % True PDF
plot(xx,pp,'r','linewidth',2)
legend('Histogram','True PDF','location','NE')
xlabel('\bf x'), ylabel('\bf Relative Frequency')
hold off
saveas(gcf,'AcceptRejectBeta102','epsc')
save('AcceptRejectBeta102','yran')
% End of program
% -------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 베타확률분포 $Beta(0.3, 0.6)$에서 난수들 1000개를 발생한다. MATLAB함수 chi2gof를 사용해서 카이제곱검정을 한 결과, 검정통계량값은 7.2184이고 자유도는 7이며 p값이 0.4065이다. 따라서 이 난수들이 베타확률분포 $Beta(0.3, 0.6)$에서 생성되었다는 귀무가설이 채택된다. 이 MATLAB프로그램을 실행하면, 그림 1.5.24이 출력된다. 그림 1.5.24에서 막대그래프는 이 난수들의 히스토그램이고, 적색 실선은 진짜확률밀도함수이다. 그림 1.5.24에서 이 난수들이 베타확률분포로부터 발생되었다는 것을 확인할 수 있다. ■

예제 1.5.25 MATLAB을 이용해서 베타확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 베타난수를 살펴보기 위해서, 다음 MATLAB프로그램 BetaRV101.m을 실행해 보자.

```
% -------------------------------------------------------------------
%  Filename: BetaRV101.m
%  Beta Random Variable
%  Programmed by CBS
```

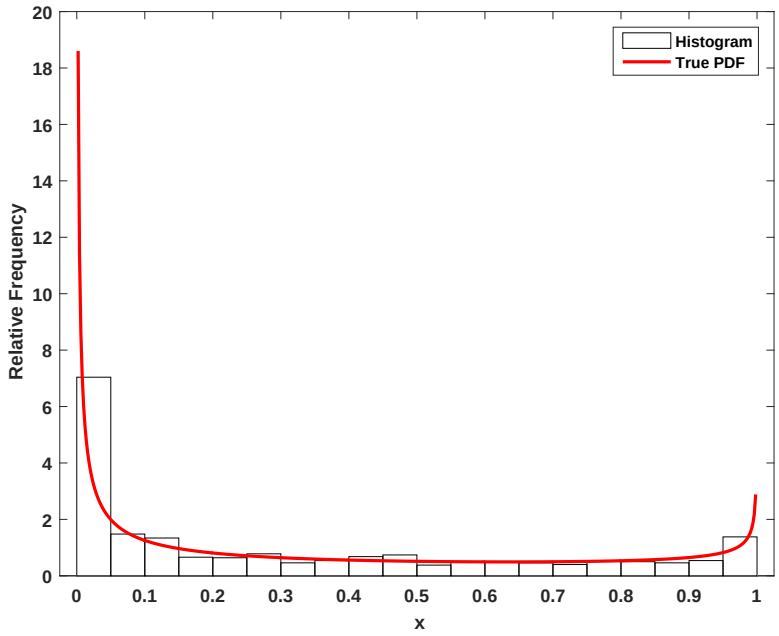


그림 1.5.24. 채택기각법과 베타확률분포

```
%-------------------------------------------------------------------
clear all, close all
a = 2, b = 5                          % Parameters
% PDF
yp1 = pdf('beta',1,a,b)
yp2 = betapdf(1,a,b)
xx = linspace(0,1,301);
yy = pdf('Beta',xx,a,b);
subplot(2,2,1)
plot(xx,yy,'r-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 1.03],'ylim',[0 3])
title('\bf PDF')
% CDF
yc1 = cdf('beta',1,a,b)
yc2 = betacdf(1,a,b)
xx = linspace(0,1,301);
yy = cdf('Beta',xx,a,b);
subplot(2,2,2)
plot(xx,yy,'g-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 1.03],'ylim',[0 1])
title('\bf CDF')
% Inverse CDF
yi1 = icdf('beta',0.5,a,b)
yi2 = betainv(0.5,a,b)
xx = 0:0.01:1;
yy = icdf('Beta',xx,a,b);
subplot(2,2,3)
plot(xx,yy,'b-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[0 1.03])
title('\bfInverse CDF')
% Random Numbers
yr1 = random('beta',a,b,3)
yr1 = betarnd(a,b,3)
rand('twister',5489)
yran = random('Beta',a,b,1,500);
subplot(2,2,4)
```

```
h1 = histogram(yran)
h1.NumBins = 20;
h1.FaceColor = [1 1 1];
h1.EdgeColor = 'black';
set(gca,'fontsize',11,'fontweigh','bold')
title('\bf Histogram')
saveas(gcf,'BetaRV101','epsc')
save('BetaRV101','yran')
% End of program
% --------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 자유도가 $(2,5)$인 베타확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 500개 베타난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.25에 그려져 있다.

베타확률분포에 관한 MATLAB함수들로는 betapdf, pdf, betacdf, cdf, betainv, icdf, betastat, betafit, mle, fitdist, dfittool, betalike, betarnd, random, randtool 등이 있다. ■

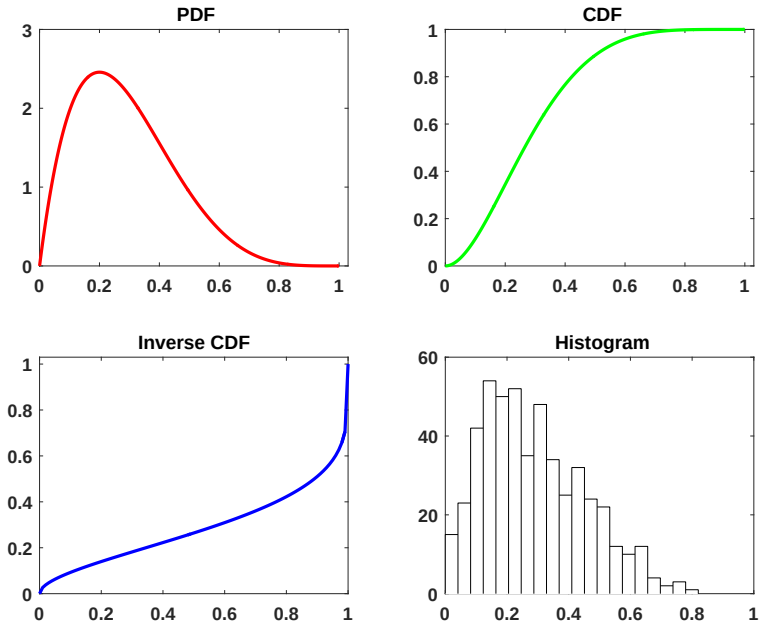


그림 1.5.25. 베타확률변수

예제 1.5.26 R을 이용해서 베타확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 베타난수를 살펴보기 위해서, 다음 R프로그램 F_RV101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: BetaRV101R.R
# Beta Random Variable
# Programmed by CBS
# ------------------------------------------------------------------------
nAbsissa <- 801
a <- 2; b <- 5
```

```
x <- seq(0,1,len=nAbsissa)
yp <- dbeta(x, shape1=a, shape2=b, ncp=0, log=FALSE)
yc <- pbeta(x, shape1=a, shape2=b, ncp=0, lower.tail=TRUE, log.p=FALSE)
ix <- seq(0,1,len=nAbsissa)
yi <- qbeta(ix, shape1=a, shape2=b, ncp=0, lower.tail=TRUE, log.p=FALSE)
nSim <- 500
set.seed(41)
yran <- rbeta(nSim, shape1=a,shape2=b, ncp=0)

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('BetaRV101R.eps')  # Start to save figure
RVdata1 <- data.frame(x,yp,yc)
RVdata2 <- data.frame(ix,yi)
RVdata3 <- data.frame(yran)
plot11 <- ggplot(RVdata1, aes(x,yp)) +
          geom_line(col="red",lwd=1.2) +
          xlab("x") +
          ggtitle("PDF")
plot12 <- ggplot(RVdata1, aes(x,yc)) +
          geom_line(col="green",lwd=1.2) +
          xlab("x") +
          ggtitle("CDF")
plot13 <- ggplot(RVdata2, aes(ix,yi)) +
          geom_line(col="blue",lwd=1.2) +
          xlab("x") +
          ggtitle("Inverse CDF")
plot14 <- ggplot(RVdata3, aes(x=yran)) +
          geom_histogram(bins=20,fill="white",color="black")+
          xlab("x") +
          ggtitle("Histogram")
pushViewport(viewport(layout=grid.layout(2,2)))
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot13, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot14, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()             # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 자유도가 $(2,5)$인 베타확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 500개 베타난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.26에 그려져 있다. ■

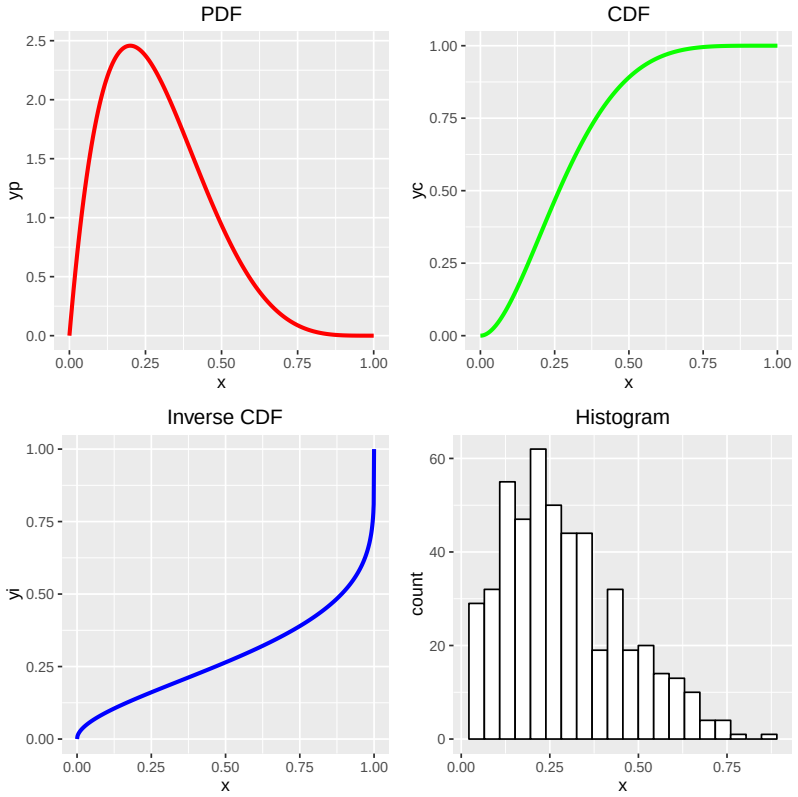


그림 1.5.26. 베타확률변수와 R

1.5.11 Weibull확률분포

척도모수(scale parameter)가 $a(>0)$이고 형태모수(shape parameter)가 $b(>0)$인 Weibull 확률분포의 확률밀도함수는 다음과 같다.

$$f(x) = \frac{b}{a}\left[\frac{x}{a}\right]^{b-1} \exp\left(-\left[\frac{x}{a}\right]^{b}\right) 1_{[0,\infty)}(x) \tag{1.5.47}$$

또한, 누적확률분포함수는 다음과 같다.

$$F(x) = \left[1 - \exp\left(-\left[\frac{x}{a}\right]^{b}\right)\right] 1_{[0,\infty)}(x) \tag{1.5.48}$$

이 Weibull확률분포의 평균과 분산은 각각 다음과 같다.

$$E(x) = \frac{a}{b}\Gamma\left(\frac{1}{b}\right), \qquad Var(x) = \frac{a^2}{b^2}\left\{2b\Gamma\left(\frac{2}{b}\right) - \left[\Gamma\left(\frac{1}{b}\right)\right]^2\right\} \tag{1.5.49}$$

만약 형태모수가 $b = 3.602$이면, 이 Weibull확률분포는 정규확률분포에 가깝다. 만약 $b > 3.602$이면, 이 Weibull확률분포는 왼쪽 꼬리가 무겁다. 만약 $b < 3.602$이면, 이 Weibull

확률분포는 오른쪽 꼬리가 무겁다. 만약 $b \leq 1$이면, Weibull확률밀도함수는 L모양(L-shaped)이다. 만약 $b > 1$이면, Weibull확률밀도함수는 종모양(bell-shaped)이다. 만약 b가 크면, Weibull확률밀도함수는 최빈값(mode) 주변에서 급격한 피크를 갖는다.

Weibull확률분포함수의 역함수는 다음과 같다.

$$F^{-1}(x) = a[-\ln(1-x)]^{1/b} 1_{[0,1)}(x) \tag{1.5.50}$$

식 (1.5.50)과 역함수법을 사용해서, Weibull난수를 생성할 수 있다.

예제 1.5.27 역함수법을 사용해서 Weibull난수들을 생성하기 위해서 다음 MATLAB프로그램 InverseMethodWeibull101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: InverseMethodWeibull101.m
%  Weibull Random Numbers by Inverse Method
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
a = 1.7, b = 4                  % Parameters
N = 1000                        % No of RN
rand('twister',5489)
yran = a*(-log(rand(1,N))).^(1/b);
% Goodness-of-Fit Test
[H,P_value,stat] = chi2gof(yran,'cdf',{@wblcdf,a,b})
% Plotting Histogram and PDF
[ Nbar,xcenter ] = hist(yran,21);
Nheight = Nbar/( xcenter(2)-xcenter(1) )/N;
bar(xcenter,Nheight,1,'w')
xdum = a/b*gamma(1/b)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[-0.1*xdum 3*xdum ])
hold on
xx = linspace(0,6,1001);
pp = wblpdf(xx,a,b);       % True PDF
plot(xx,pp,'r','linewidth',2)
legend('Histogram','True PDF','location','NE')
xlabel('\bf x'), ylabel('\bf Relative Frequency')
hold off
saveas(gcf,'InverseMethodWeibull101','epsc')
save('InverseMethodWeibull101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 척도모수가 $a = 1.7$이고 형태모수가 $b = 4$인 Weibull확률분포에서 난수들 1000개를 발생한다. MATLAB함수 chi2gof를 사용해서 카이제곱검정을 한 결과, 검정통계량값은 10.3728이고 자유도는 7이며 p값이 0.1684이다. 따라서 이 난수들이 Weibull확률분포에서 생성되었다는 귀무가설이 채택된다. 이 MATLAB프로그램을 실행하면, 그림 1.5.18이 출력된다. 그림 1.5.18에서 막대그래프는 이 난수들의 히스토그램이고,

적색 실선은 진짜확률밀도함수이다. 그림 1.5.27에서 이 난수들이 Weibull확률분포로부터 발생되었다는 것을 확인할 수 있다. ■

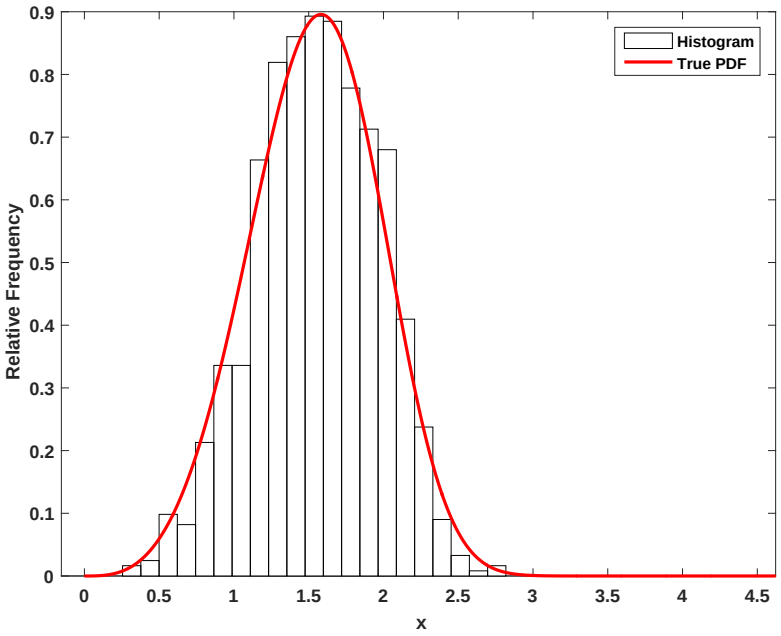


그림 1.5.27. 역함수법과 Weibull확률분포

예제 1.5.28 MATLAB을 이용해서 Weibull확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 Weibull난수를 살펴보기 위해서, 다음 MATLAB프로그램 WeibullRV101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: WeibullRV101.m
%  Weibull Random Variable
%  Programmed by CBS
%-------------------------------------------------------------------------
clear all, close all
a = 2, b = 5                          % Parameters
% PDF
yp1 = pdf('wbl',1,a,b)
yp2 = wblpdf(1,a,b)
xx = linspace(0,5,1001);
yy = pdf('wbl',xx,a,b);
subplot(2,2,1)
plot(xx,yy,'r-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 3.33],'ylim',[0 1])
title('\bf PDF')
% CDF
yc1 = cdf('wbl',1,a,b)
yc2 = wblcdf(1,a,b)
xx = linspace(0,5,1001);
yy = cdf('wbl',xx,a,b);
subplot(2,2,2)
plot(xx,yy,'g-','linewidth',2)
```

```
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 3.33],'ylim',[0 1])
title('\bf CDF')
% Inverse CDF
yi1 = icdf('wbl',0.5,a,b)
yi2 = wblinv(0.5,a,b)
xx = 0:0.01:1;
yy = icdf('wbl',xx,a,b);
subplot(2,2,3)
plot(xx,yy,'b-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[0 3.33])
title('\bfInverse CDF')
% Random Numbers
yr1 = random('wbl',a,b,3)
yr1 = wblrnd(a,b,3)
rand('twister',5489)
yran = random('wbl',a,b,1,500);
subplot(2,2,4)
h1 = histogram(yran)
h1.NumBins = 20;
h1.FaceColor = [1 1 1];
h1.EdgeColor = 'black';
set(gca,'fontsize',11,'fontweigh','bold')
title('\bf Histogram')
saveas(gcf,'WeibullRV101','epsc')
save('WeibullRV101','yran')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 척도모수가 $a = 2$이고 형태모수가 $b = 5$인 Weibull 확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 500개 Weibull난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.28에 그려져 있다.

Weibull확률분포에 관한 MATLAB함수들로는 wblpdf, pdf, wblcdf, cdf, wblinv, icdf, wblstat, wblfit, mle, fitdist, dfittool, wbllike, wblrnd, random 등이 있다. ■

예제 1.5.29 R을 이용해서 Weibull확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 Weibull난수를 살펴보기 위해서, 다음 R프로그램 WeibullRV101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: WeibullRV101R.R
# Weibull Random Variable
# Programmed by CBS
# ------------------------------------------------------------------------
nAbsissa <- 801
scaleA <- 2; shapeB <- 5
x <- seq(0,3.33,len=nAbsissa)
yp <- dweibull(x, shape=shapeB, scale=scaleA, log=FALSE)
yc <- pweibull(x, shape=shapeB, scale=scaleA, lower.tail=TRUE, log.p=FALSE)
ix <- seq(0,1,len=nAbsissa)
yi <- qweibull(ix, shape=shapeB, scale=scaleA, lower.tail=TRUE, log.p=FALSE)
```

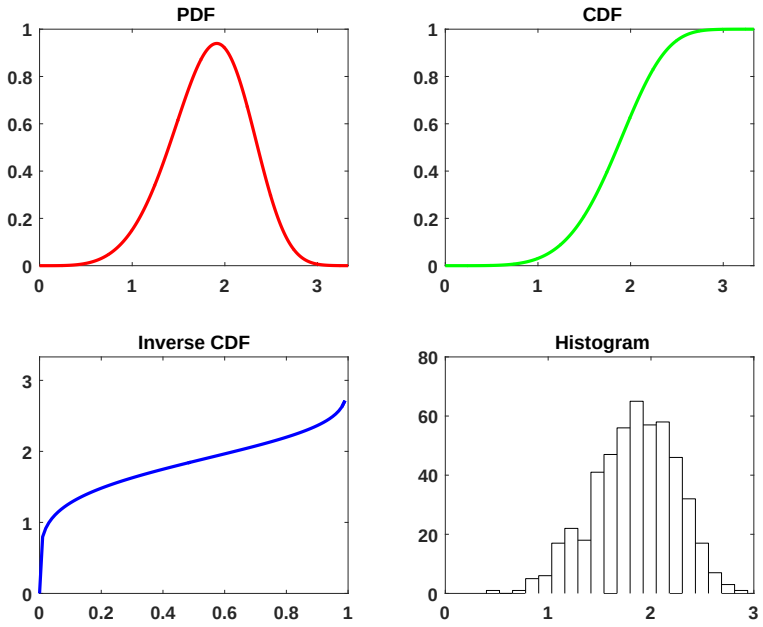


그림 1.5.28. Weibull확률변수와 MATLAB

```
nSim <- 500
set.seed(41)
yran <- rweibull(nSim, shape=shapeB, scale=scaleA)

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('WeibullRV101R.eps')  # Start to save figure
RVdata1 <- data.frame(x,yp,yc)
RVdata2 <- data.frame(ix,yi)
RVdata3 <- data.frame(yran)
plot11 <- ggplot(RVdata1, aes(x,yp)) +
          geom_line(col="red",lwd=1.2) +
          xlab("x") +
          ggtitle("PDF")
plot12 <- ggplot(RVdata1, aes(x,yc)) +
          geom_line(col="green",lwd=1.2) +
          xlab("x") +
          ggtitle("CDF")
plot13 <- ggplot(RVdata2, aes(ix,yi)) +
          geom_line(col="blue",lwd=1.2) +
          xlab("x") +
          ggtitle("Inverse CDF")
plot14 <- ggplot(RVdata3, aes(x=yran)) +
          geom_histogram(bins=20,fill="white",color="black")+
          xlab("x") +
          ggtitle("Histogram")
pushViewport(viewport(layout=grid.layout(2,2)))
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot13, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot14, vp=viewport(layout.pos.row=2, layout.pos.col=2))
```

```
49 dev.off()              # End to save figure
50 # ---------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 척도모수가 2이고 형태모수가 5인 Weibull확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 여기서 유의할 점은 MATLAB에서와는 달리 R에서는 형태모수를 a, 그리고 척도모수를 b로 표기한다는 것이다. 또한, 이 확률분포에서 500개 Weibull난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.29에 그려져 있다. ■

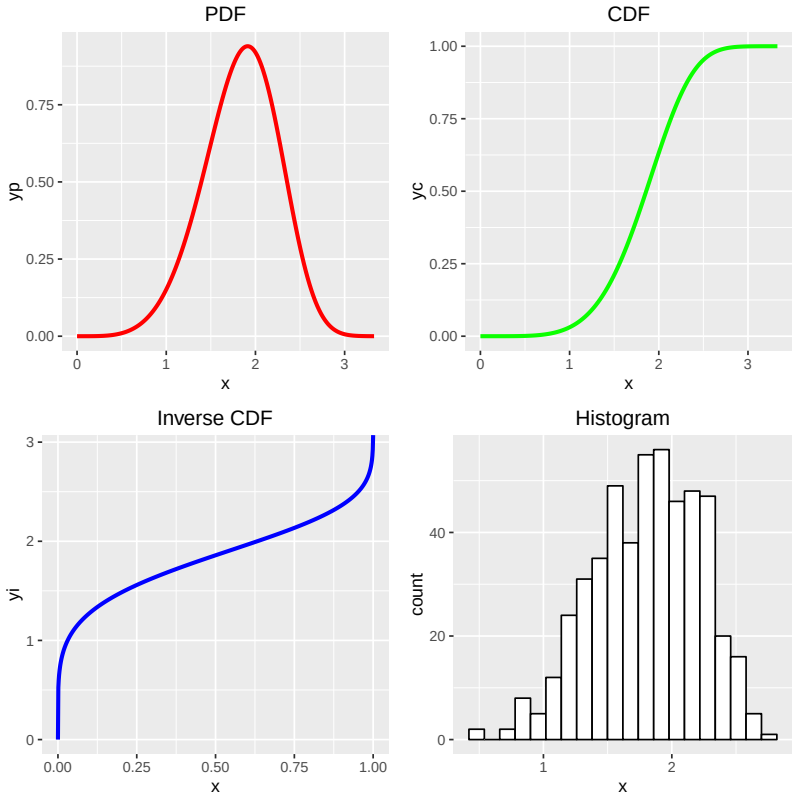


그림 1.5.29. Weibull확률변수와 R

1.5.12 Rayleigh확률분포

척도모수(scale parameter)가 b인 Rayleigh확률분포의 확률밀도함수는 다음과 같다.

$$f(x) = \frac{x}{b^2} \exp\left(-\frac{x^2}{2b^2}\right) 1_{[0,\infty)}(x) \tag{1.5.51}$$

식 (1.5.47) 에서 알 수 있듯이, Rayleigh확률분포는 Weibull확률분포의 특수한 경우이다. Rayleigh확률분포함수는 다음과 같다.

$$F(x) = \left[1 - \exp\left(-\frac{x^2}{2b^2}\right)\right] 1_{[0,\infty)}(x) \tag{1.5.52}$$

이 Rayleigh확률분포의 평균과 분산은 각각 다음과 같다.

$$E(x) = \sqrt{\frac{\pi}{2}}\sigma, \qquad Var(x) = \frac{4-\pi}{2}b^2 \tag{1.5.53}$$

Rayleigh확률분포함수의 역함수는 다음과 같다.

$$F^{-1}(x) = b\sqrt{-2\ln(1-x)}1_{[0,1)}(x) \tag{1.5.54}$$

식 (1.5.54) 와 역함수법을 사용해서, Rayleigh난수들을 생성할 수 있다.

예제 1.5.30 MATLAB을 이용해서 Rayleigh확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 Rayleigh난수를 살펴보기 위해서, 다음 MATLAB프로그램 RayleighRV101.m을 실행해 보자

```
% ------------------------------------------------------------------
%  Filename: RayleighRV101.m
%  Rayleigh Random Variable
%  Programmed by CBS
%-------------------------------------------------------------------
clear all, close all
b = 2                                  % Parameters
% PDF
yp1 = pdf('rayl',1,b)
yp2 = raylpdf(1,b)
xx = 0:0.05:10;
yy = pdf('Rayleigh',xx,b);
subplot(2,2,1)
plot(xx,yy,'r-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold', ...
    'xlim',[0 8.1],'ylim',[0 0.35])
title('\bf PDF')
% CDF
yc1 = cdf('rayl',1,b)
yc2 = raylcdf(1,b)
xx = 0:0.05:10;
yy = cdf('Rayleigh',xx,b);
subplot(2,2,2)
plot(xx,yy,'g-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold', ...
    'xlim',[0 8.1],'ylim',[0 1])
title('\bf CDF')
% Inverse CDF
```

```
yi1 = icdf('rayl',0.5,b)
yi2 = raylinv(0.5,b)
xx = 0:0.01:1;
yy = icdf('Rayleigh',xx,b);
subplot(2,2,3)
plot(xx,yy,'b-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[0 8.1])
title('\bfInverse CDF')
% Random Numbers
yr1 = random('rayl',b,3)
yr1 = raylrnd(b,3)
rand('twister',5489)
yran = random('Rayleigh',b,1,500);
subplot(2,2,4)
h1 = histogram(yran)
h1.NumBins = 20;
h1.FaceColor = [1 1 1];
h1.EdgeColor = 'black'
set(gca,'fontsize',11,'fontweigh','bold')
title('\bf Histogram')
saveas(gcf,'RayleighRV101','epsc')
save('RayleighRV101','yran')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 척도모수가 $b = 2$인 Rayleigh확률변수의 확률밀도함수, 누적확률분포함수와 역누적확률분포함수를 그린다. 또한 이 MATLAB프로그램을 실행하면, 이 확률분포에서 500개 Rayleigh난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.30에 그려져 있다.

Rayleigh확률분포에 관한 MATLAB함수들로는 raylpdf, pdf, raylcdf, cdf, raylinv, icdf, raylstat, raylfit, mle, fitdist, dfittool, raylrnd, random, randtool 등이 있다. ■

예제 1.5.31 R패키지 VGAM을 이용해서 Rayleigh확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 Rayleigh난수를 살펴보기 위해서, 다음 R프로그램 RayleighRV101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename RayleighRV101R.R
# Rayleigh Random Variable
# Programmed by CBS
# ------------------------------------------------------------------------
# install.packages("VGAM")
library(VGAM)
b <- 2
nAbsissa <- 801
x <- seq(0,8,len=nAbsissa)
yp <- drayleigh(x, scale=b, log=FALSE)   # PDF
yc <- prayleigh(x, scale=b, lower.tail=TRUE, log.p=FALSE)  # CDF
ix <- seq(0,1,len=nAbsissa)
yi <- qrayleigh(ix, scale=b, log.p=FALSE) #Inverse CDF
nSim <- 500
```

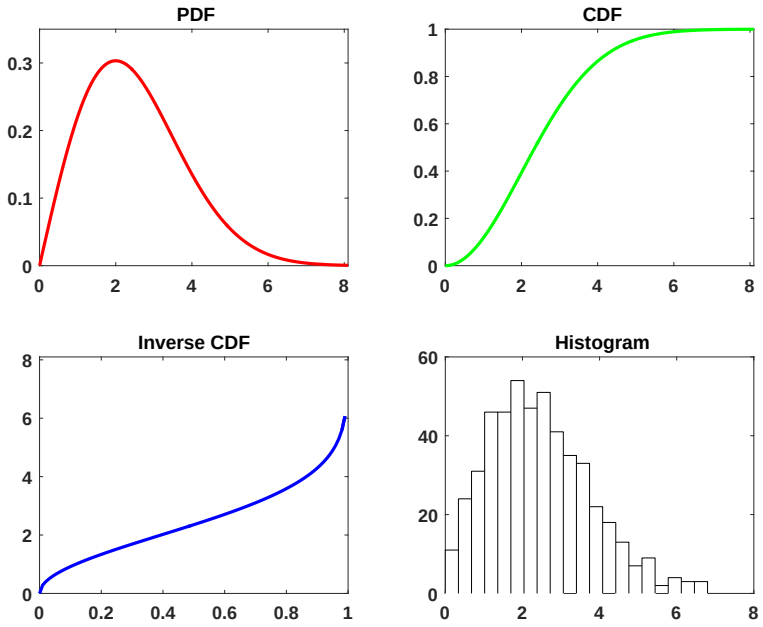


그림 1.5.30. Rayleigh확률변수

```
set.seed(41)
yran <- rrayleigh(nSim, scale=b)

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('RayleighRV101R.eps')  # Start to save figure
RVdata1 <- data.frame(x,yp,yc)
RVdata2 <- data.frame(ix,yi)
RVdata3 <- data.frame(yran)
plot11 <- ggplot(RVdata1, aes(x,yp)) +
          geom_line(col="red",lwd=1.2) +
          xlab("x") +
          ggtitle("PDF")
plot12 <- ggplot(RVdata1, aes(x,yc)) +
          geom_line(col="green",lwd=1.2) +
          xlab("x") +
          ggtitle("CDF")
plot13 <- ggplot(RVdata2, aes(ix,yi)) +
          geom_line(col="blue",lwd=1.2) +
          xlab("x") +
          ggtitle("Inverse CDF")
plot14 <- ggplot(RVdata3, aes(x=yran)) +
          geom_histogram(bins=20,fill="white",color="black")+
          xlab("x") +
          ggtitle("Histogram")
pushViewport(viewport(layout=grid.layout(2,2)))
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot13, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot14, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()             # End to save figure
```

```
52 # ------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 척도모수가 $b = 2$인 Rayleigh확률변수의 확률밀도함수, 누적확률분포함수와 역누적확률분포함수를 그린다. 또한 이 R프로그램을 실행하면, 이 확률분포에서 500개 Rayleigh난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.31에 그려져 있다. ■

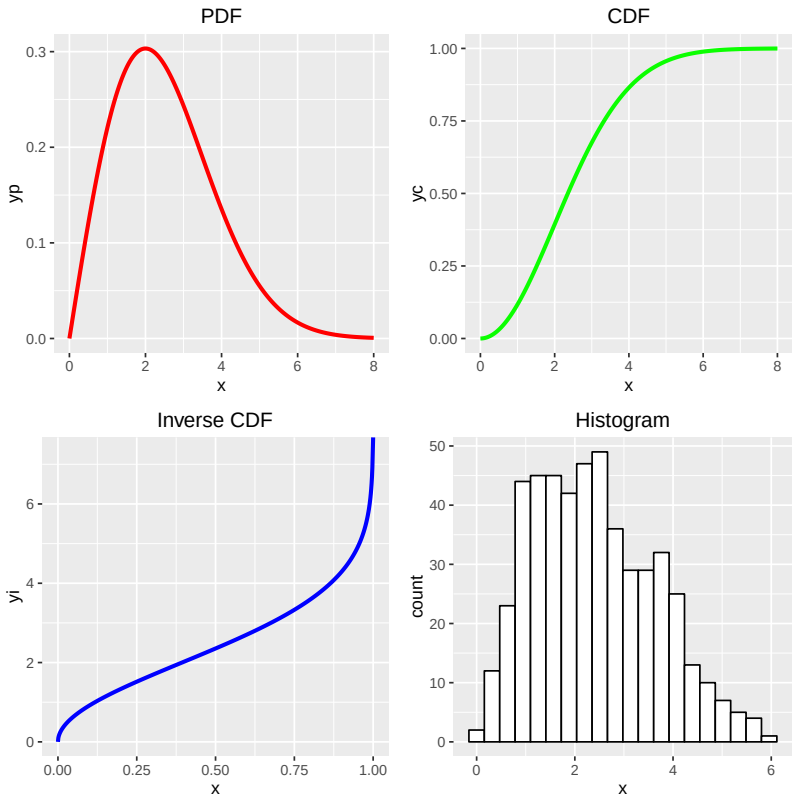


그림 1.5.31. Rayleigh확률변수와 R

1.5.13 일반화Pareto확률분포

형태모수(shape parameter)가 $k(\neq 0)$이고 위치모수(location parameter)가 θ그 리고 척도모수(scale parameter)가 $\sigma(> 0)$인 일반화Pareto확률분포(generalized Pareto probability distribution)의 확률밀도함수는 다음과 같다.

$$f(x) = \frac{1}{\sigma}\left\{1 + k\left[\frac{x-\theta}{\sigma}\right]\right\}^{-1-1/k} = \frac{\sigma^{\frac{1}{k}}}{\{\sigma + k[x-\theta]\}^{\frac{1}{k}+1}} \tag{1.5.55}$$

만약 $k > 0$ 이면, f(x)는 구간 $x > \theta$ 에서 정의된다. 만약 $k < 0$ 이면, $f(x)$ 는 구간 $\theta < x < -\sigma/k$ 에서 정의된다. 만약 $k > 0$ 이고 $\theta = \sigma/k$ 이면, 일반화Pareto확률분포는 Pareto 확률분포이다. 일반화Pareto확률분포함수는 다음과 같다.

$$F(x) = 1 - \left\{1 + k\left[\frac{x-\theta}{\sigma}\right]\right\}^{-1/k}, \quad (k \neq 0) \tag{1.5.56}$$

이 일반화Pareto확률분포의 평균과 분산은 각각 다음과 같다.

$$E(x) = \theta + \frac{\sigma}{1-k}, \quad Var(x) = \frac{\sigma^2}{[k-1]^2[1-2k]} \quad \left(k < \frac{1}{2}\right) \tag{1.5.57}$$

일반화Pareto확률분포함수의 역함수는 다음과 같다.

$$F^{-1}(x) = \theta + \frac{\sigma}{k}\left\{[1-x]^{-k} - 1\right\} \tag{1.5.58}$$

식 (1.5.58)과 역함수법을 사용해서, 일반화Pareto난수를 생성할 수 있다.

예제 1.5.32 역함수법을 사용해서 일반화Pareto난수들을 생성하기 위해서 다음 MATLAB 프로그램 InverseMethodGeneralizedPareto101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: InverseMethodGeneralizedPareto101.m
%  Generating Generalized Pareto RN by Inverse Method
%-------------------------------------------------------------------
clear all, close all
k = 1/3, theta = 1, sigma = 2   % Parameters
N = 1000
rand('twister',5489)
yran = theta + sigma/k*(rand(1,N).^(-k)-1);
[Nbar,xcenter ] = hist(yran,31);
IntWidth = xcenter(2)-xcenter(1);
bar(xcenter,Nbar/N/IntWidth,1)
set(get(gca,'child'),'FaceColor',[0.9 0.9 0.9])
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 30])
xx = linspace(1+eps,45+eps,201);
line(xx,gppdf(xx,k,sigma,theta),'linewidth',2);
legend('\bf Histogram','\bf Estimated PDF',1)
saveas(gcf,'InverseMethodGeneralizedPareto101','epsc')
save('InverseMethodGeneralizedPareto101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 형태모수가 $k = 1/3$ 이고 위치모수가 $\theta = 1$ 그리고 척도모수가 $\sigma = 2$ 인 일반화Pareto확률분포에서 난수들 1000 개를 발생한다. 이 MATLAB 프로그램을 실행하면, 그림 1.5.32 이 출력된다. 그림 1.5.32 에서 막대그래프는 이 난수들의

히스토그램이고, 청색 실선은 진짜확률밀도함수이다. 그림 1.5.32 에서 이 난수들이 일반화 Pareto확률분포로부터 발생되었다는 것을 알 수 있다. ■

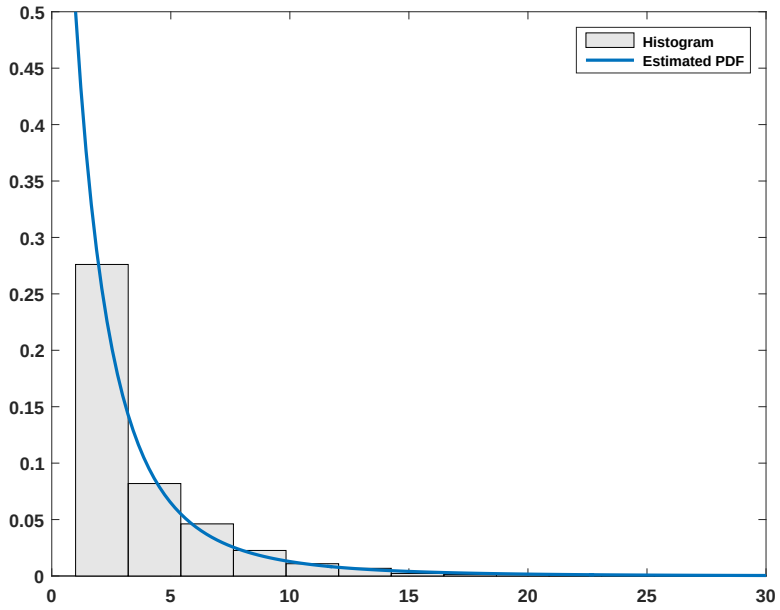


그림 1.5.32. 역함수법과 일반화Pareto확률분포

예제 1.5.33 MATLAB을 이용해서 일반화Pareto확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 일반화Pareto난수를 살펴보기 위해서, 다음 MATLAB 프로그램 GeneralizedParetoRV101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: GeneralizedParetoRV101.m
%  Generalized Pareto Random Variable
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
k = 1/3, theta = 1, sigma = 2                    % Parameters
% PDF
yp1 = pdf('gp',1,k,sigma,theta)
yp2 = gppdf(1,k,sigma,theta)
xx = linspace(0,30,301);
yy = pdf('Generalized Pareto',xx,k,sigma,theta);
subplot(2,2,1)
plot(xx,yy,'r-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold', ...
    'xlim',[0 20.12],'ylim',[0 0.55])
title('\bf PDF')
% CDF
yc1 = cdf('gp',1,k,sigma,theta)
yc2 = gpcdf(1,k,sigma,theta)
xx = linspace(0,30,301);
yy = cdf('Generalized Pareto',xx,k,sigma,theta);
subplot(2,2,2)
plot(xx,yy,'g-','linewidth',2)
```

```
set(gca,'fontsize',11,'fontweigh','bold', ...
    'xlim',[0 20.12],'ylim',[0 1.05])
title('\bf CDF')
% Inverse CDF
yi1 = icdf('gp',0.5,k,sigma,theta)
yi2 = gpinv(0.5,k,sigma,theta)
xx = 0:0.01:1;
yy = icdf('Generalized Pareto',xx,k,sigma,theta);
subplot(2,2,3)
plot(xx,yy,'b-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[0 20.12 ])
title('\bfInverse CDF')
% Random Numbers
yr1 = random('gp',k,sigma,theta,3)
yr1 = gprnd(k,sigma,theta,3)
rand('twister',5489)
yran = random('Generalized Pareto',k,sigma,theta,1,500);
subplot(2,2,4)
h1 = histogram(yran)
set(gca,'fontsize',11,'fontweigh','bold')
h1.NumBins = 20;
h1.FaceColor = [1 1 1];
h1.EdgeColor = 'black';
title('\bf Histogram')
saveas(gcf,'GeneralizedParetoRV101','epsc')
save('GeneralizedParetoRV101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 형태모수가 $k = 1/3$이고 위치모수가 $\theta = 1$그리고 척도모수가 $\sigma = 2$인 일반화Pareto확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 500개 일반화Pareto난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.33에 그려져 있다.

일반화Pareto확률분포에 관한 MATLAB함수들로는 gppdf, pdf, gpcdf, cdf, gpinv, icdf, gpstat, gpfit, mle, fitdist, dfittool, gplike, gprnd, random, randtool 등이 있다. ■

예제 1.5.34 R패키지 fExtremes를 이용해서 일반화Pareto확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 일반화Pareto난수를 살펴보기 위해서, 다음 R 프로그램 GeneralizedParetoRV101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename GeneralizedParetoRV101R.R
# Generalized Pareto Random Variable
# Programmed by CBS
# ------------------------------------------------------------------------
# install.packages("fExtremes")
library(fExtremes)
k <- 1/3; theta <- 1 ; sigma <- 2
b <- 2
nAbsissa <- 801
x <- seq(0,20,len=nAbsissa)
```

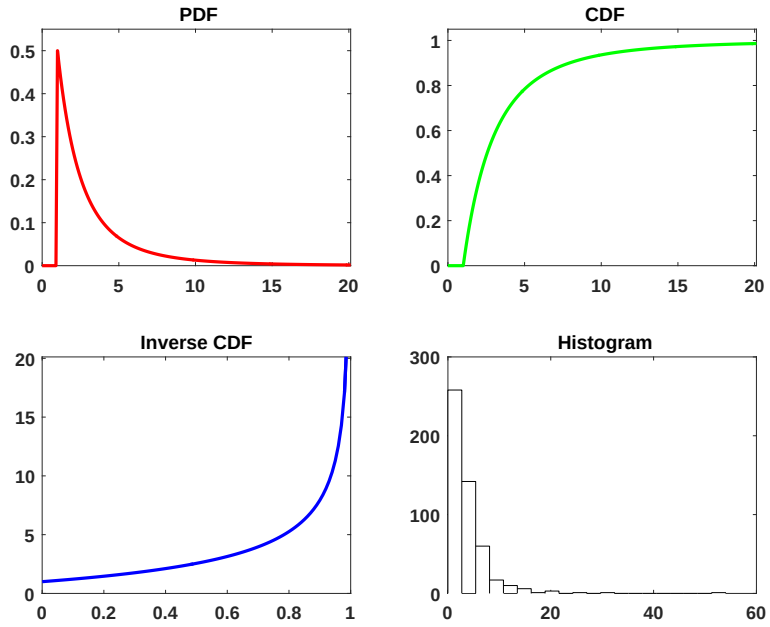


그림 1.5.33. 일반화Pareto확률분포와 MATLAB

```
yp <- dgpd(x, xi=k, mu=theta, beta=sigma, log=FALSE)   # PDF
yc <- pgpd(x, xi=k, mu=theta, beta=sigma, lower.tail=TRUE)  # CDF
ix <- seq(0,1,len=nAbsissa)
yi <- qgpd(ix, xi=k, mu=theta, beta=sigma, lower.tail=TRUE) #Inverse CDF
nSim <- 500
set.seed(41)
yran <- rgpd(x, xi=k, mu=theta, beta=sigma)

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('GeneralizedParetoRV101R.eps')  # Start to save figure
RVdata1 <- data.frame(x,yp,yc)
RVdata2 <- data.frame(ix,yi)
RVdata3 <- data.frame(yran)
plot11 <- ggplot(RVdata1, aes(x,yp)) +
          geom_line(col="red",lwd=1.2) +
          xlab("x") +
          ggtitle("PDF")
plot12 <- ggplot(RVdata1, aes(x,yc)) +
          geom_line(col="green",lwd=1.2) +
          xlab("x") +
          ggtitle("CDF")
plot13 <- ggplot(RVdata2, aes(ix,yi)) +
          geom_line(col="blue",lwd=1.2) +
          xlab("x") +
          ggtitle("Inverse CDF")
plot14 <- ggplot(RVdata3, aes(x=yran)) +
          geom_histogram(bins=20,fill="white",color="black")+
          xlab("x") +
          ggtitle("Histogram")
pushViewport(viewport(layout=grid.layout(2,2)))
```

```
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot13, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot14, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()             # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 척도모수가 $b = 2$인 일반Pareto확률변수의 확률밀도함수, 누적확률분포함수와 역누적확률분포함수를 그린다. 또한 이 R프로그램을 실행하면, 이 확률분포에서 500개 GeneralizedPareto난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.34에 그려져 있다. ■

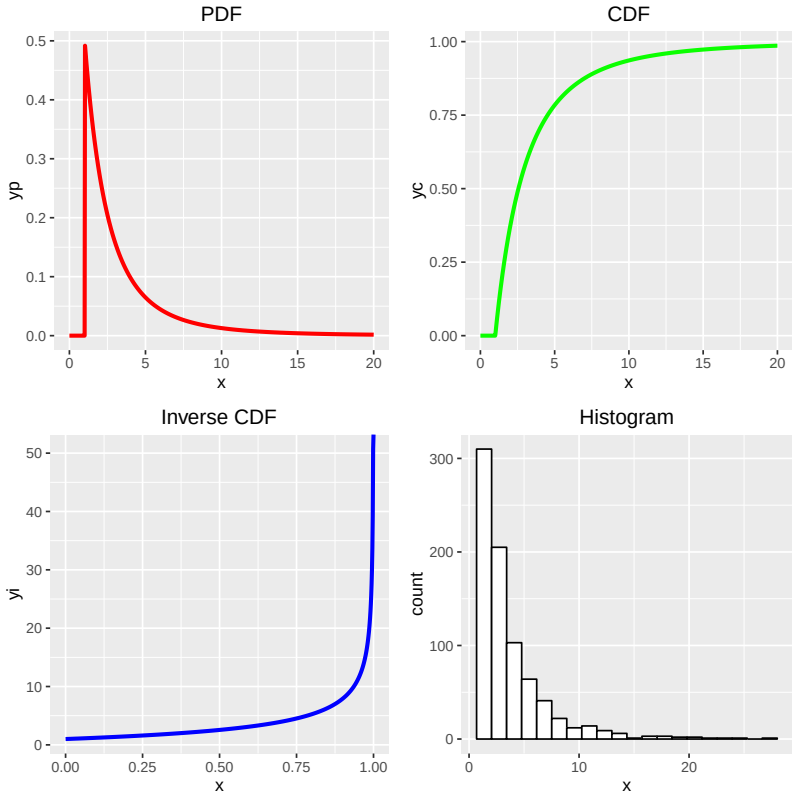


그림 1.5.34. 일반화Pareto확률변수와 R

1.5.14 일반화극한값확률분포

형태모수가 $k \neq 0$이고 위치모수가 θ그리고 척도모수가 $\sigma(>0)$인 일반화극한값확률분포(generalized extreme value probability distribution)의 확률밀도함수는 다음과 같다.

$$f(x) = \frac{1}{\sigma}\left\{1+k\left[\frac{x-\theta}{\sigma}\right]\right\}^{-1-1/k} \exp\left(-\left\{1+k\left[\frac{x-\theta}{\sigma}\right]\right\}\right) 1\left(1+k\left[\frac{x-\theta}{\sigma}\right]>0\right) \tag{1.5.59}$$

이에 해당하는 누적확률분포함수는 다음과 같다.

$$F(x) = \exp\left(-\left\{1+k\left[\frac{x-\theta}{\sigma}\right]\right\}^{-1/k}\right) 1\left(1+k\left[\frac{x-\theta}{\sigma}\right]>0\right) \tag{1.5.60}$$

또한, 평균과 분산은 각각 다음과 같다.

$$E(x) = \theta - \frac{\sigma}{k} + \frac{\sigma}{k}g_1, \quad Var(x) = \frac{\sigma^2}{k^2}\left[g_2 - g_1^2\right] \tag{1.5.61}$$

여기서 $g_j \doteq \Gamma(1+jk),\ (j=1,2)$이다.

만약 $k=0$이면, 일반화극한값확률분포의 확률밀도함수는 다음과 같다.

$$f(x) = \frac{1}{\sigma}\exp\left(-\exp\left(-\frac{x-\theta}{\sigma}\right) - \frac{x-\theta}{\sigma}\right) \tag{1.5.62}$$

식 (1.5.62)의 확률밀도함수는 식 (1.5.59)의 확률밀도함수에 극한 $k \to 0$을 취한 결과와 같다. 이에 해당하는 확률분포를 극한값확률분포라 부른다. 극한값확률분포함수는 다음과 같다.

$$F(x) = \exp\left(-\exp\left(-\frac{x-\theta}{\sigma}\right)\right) \tag{1.5.63}$$

또한, 평균과 분산은 각각 다음과 같다.

$$E(x) = \theta + \gamma\sigma \quad Var(x) = \frac{1}{6}\pi^2\sigma^2 \tag{1.5.64}$$

여기서 γ는 Euler상수이다.

예제 1.5.35 MATLAB을 이용해서 일반화극한값확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 일반화극한값난수를 살펴보기 위해서, 다음 MATLAB 프로그램 GeneralizedExtremeValueRV101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: GeneralizedExtremeValueRV101.m
%  Generalized Exetreme Value Random Variable
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
k = 1/3, theta = 1, sigma = 2                          % Parameters
% PDF
yp1 = pdf('gev',1,k,sigma,theta)
yp2 = gevpdf(1,k,sigma,theta)
xx = linspace(-5,30,301);
yy = pdf('gev',xx,k,sigma,theta);
subplot(2,2,1)
plot(xx,yy,'r-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold', ...
    'xlim',[-2.2 20.1],'ylim',[0 .25])
title('\bf PDF')
% CDF
yc1 = cdf('gev',1,k,sigma,theta)
yc2 = gevcdf(1,k,sigma,theta)
xx = linspace(-5,30,301);
yy = cdf('gev',xx,k,sigma,theta);
subplot(2,2,2)
plot(xx,yy,'g-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold', ...
    'xlim',[-2.2 20.1 ],'ylim',[0 1])
title('\bf CDF')
% Inverse CDF
yi1 = icdf('gev',0.5,k,sigma,theta)
yi2 = gevinv(0.5,k,sigma,theta)
xx = 0:0.01:1;
yy = icdf('gev',xx,k,sigma,theta);
subplot(2,2,3)
plot(xx,yy,'b-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[-2.2 20.1 ])
title('\bfInverse CDF')
% Random Numbers
rand('twister',5489)
yr1 = random('gev',k,sigma,theta,3)
yr1 = gevrnd(k,sigma,theta,3)
yran = random('gev',k,sigma,theta,1,500);
subplot(2,2,4)
h1 = histogram(yran)
set(gca,'fontsize',11,'fontweigh','bold')
h1.NumBins = 20;
h1.FaceColor = [1 1 1];
h1.EdgeColor = 'black';
title('\bf Histogram')
% If k = 0, then we can use 'ev' instead of 'gev'.
saveas(gcf,'GeneralizedExtremeValueRV101','epsc')
save('GeneralizedExtremeValueRV101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 형태모수가 $k = 1/3$이고 위치모수가 $\theta = 1$그리고 척도모수가 $\sigma = 2$인 일반화극한값확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 난수들을 100개 생성하고, 이들의 히스토그램을

그린다. 이 그래프들이 그림 1.5.35에 그려져 있다.

일반화극한값확률분포에 관한 MATLAB함수들로는 gevpdf, pdf, gevcdf, cdf, gevinv, icdf, gevstat, gevfit, mle, fitdist, dfittool, gevlike, gevrnd, random, randtool 등이 있다. ■

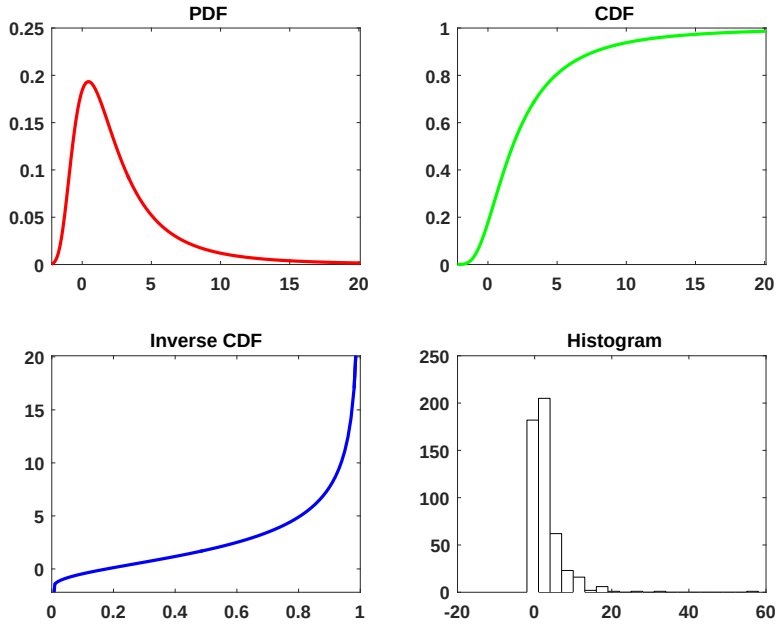


그림 1.5.35. 일반화극한값확률분포

예제 1.5.36 R패키지 fExtremes를 이용해서 일반화극한값확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 일반화극한값난수를 살펴보기 위해서, 다음 R 프로그램 GeneralizedExtremeValueRV101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------------
# Filename GeneralizedExtremeValueRV101R.R
# Generalized Extreme Value Random Variable
# Programmed by CBS
# ------------------------------------------------------------------------------
# install.packages("fExtremes")
library(fExtremes)
k <- 1/3; theta <- 1 ; sigma <- 2
b <- 2
nAbsissa <- 801
x <- seq(-2.2,20.1,len=nAbsissa)
yp <- dgev(x, xi=k, mu=theta, beta=sigma, log=FALSE)   # PDF
yc <- pgev(x, xi=k, mu=theta, beta=sigma, lower.tail=TRUE)  # CDF
ix <- seq(0,1,len=nAbsissa)
yi <- qgev(ix, xi=k, mu=theta, beta=sigma, lower.tail=TRUE) #Inverse CDF
nSim <- 500
set.seed(41)
yran <- rgev(x, xi=k, mu=theta, beta=sigma)

# Plotting
# install.packages("ggplot2")
library(ggplot2)
```

```
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('GeneralizedExtremeValueRV101R.eps')  # Start to save figure
RVdata1 <- data.frame(x,yp,yc)
RVdata2 <- data.frame(ix,yi)
RVdata3 <- data.frame(yran)
plot11 <- ggplot(RVdata1, aes(x,yp)) +
          geom_line(col="red",lwd=1.2) +
          xlab("x") +
          ggtitle("PDF")
plot12 <- ggplot(RVdata1, aes(x,yc)) +
          geom_line(col="green",lwd=1.2) +
          xlab("x") +
          ggtitle("CDF")
plot13 <- ggplot(RVdata2, aes(ix,yi)) +
          geom_line(col="blue",lwd=1.2) +
          xlab("x") +
          ggtitle("Inverse CDF")
plot14 <- ggplot(RVdata3, aes(x=yran)) +
          geom_histogram(bins=20,fill="white",color="black")+
          xlab("x") +
          ggtitle("Histogram")
pushViewport(viewport(layout=grid.layout(2,2)))
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot13, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot14, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()            # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 척도모수가 $b = 2$인 일반화극한값확률변수의 확률밀도함수, 누적확률분포함수와 역누적확률분포함수를 그린다. 또한 이 R프로그램을 실행하면, 이 확률분포에서 500개 일반화극한값난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.36에 그려져 있다. ■

1.5.15 비중심카이제곱확률분포

서로 독립인 확률변수들 $v_1, v_2, \cdots, v_\nu$가 각각 다음과 같은 확률분포를 따른다고 하자.

$$v_i \overset{d}{\sim} N(\mu_i, \sigma_i^2), \quad (i = 1, 2, \cdots, \nu) \tag{1.5.65}$$

다음과 같은 확률변수와 상수를 정의하자.

$$x \doteq \sum_{i=1}^{\nu} \left[\frac{v_i}{\sigma_i}\right]^2, \quad \delta \doteq \sum_{i=1}^{\nu} \left[\frac{v_i}{\sigma_i}\right]^2 \tag{1.5.66}$$

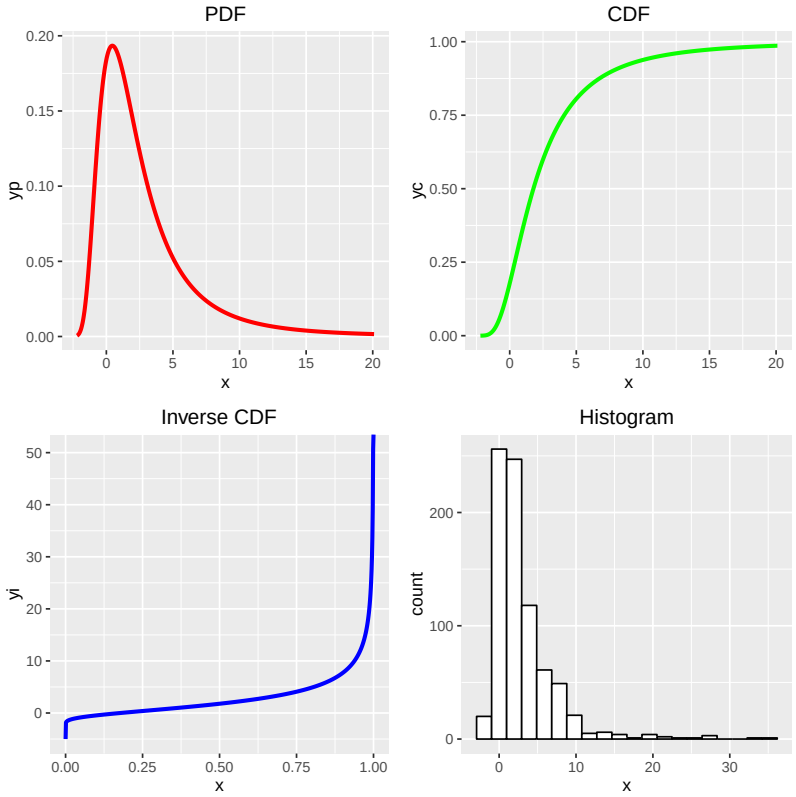


그림 1.5.36. GeneralizedExtremeValue확률변수와 R

확률변수 x는 자유도가 ν이고 비중심모수(noncentrality)가 δ인 비중심카이제곱확률분포(noncentral chi-square probability distribution) $\chi^2_\nu(\delta)$를 따른다고 한다. 문헌에 따라서는 $\delta/2$를 비중심모수라고 부르기도 한다. 이 비중심카이제곱확률변수의 확률밀도함수는 다음과 같다.

$$f(x) = \frac{1}{2}\left[\frac{x}{\delta}\right]^{[\nu-2]/4} \exp\left(-\frac{x+\delta}{2}\right) I_{[\nu-2]/2}(\sqrt{\delta x}) \tag{1.5.67}$$

여기서 제1종 변형Bessel함수 $I_a(y)$는 다음과 같다.

$$I_a(y) \doteq \left[\frac{y}{2}\right]^a \sum_{j=0}^{\infty} \frac{1}{j!\Gamma(a+j+1)} \left[\frac{y^2}{4}\right]^j \tag{1.5.68}$$

이 비중심카이제곱확률분포의 평균과 분산은 각각 다음과 같다.

$$E(x) = \nu + \delta, \quad Var(x) = 2\nu + 4\delta \tag{1.5.69}$$

예제 1.5.37 MATLAB을 이용해서 비중심카이제곱확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 비중심카이제곱난수를 살펴보기 위해서, 다음 MATLAB

프로그램 NoncentralChi2RV101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: NoncentralChi2RV101.m
%  Noncentral Chi-Squared Random Variable
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
nu = 9, delta = 3.3                                % Parameters
% PDF
yp1 = pdf('ncx2',1,nu,delta)
yp2 = ncx2pdf(1,nu,delta)
xx = 0:0.05:30;
yy = pdf('ncx2',xx,nu,delta);
subplot(2,2,1)
plot(xx,yy,'r-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold', ...
    'xlim',[0 30.12],'ylim',[0 0.08])
title('\bf PDF')
% CDF
yc1 = cdf('ncx2',1,nu,delta)
yc2 = ncx2cdf(1,nu,delta)
xx = 0:0.05:30;
yy = cdf('ncx2',xx,nu,delta);
subplot(2,2,2)
plot(xx,yy,'g-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold', ...
    'xlim',[0 30.12],'ylim',[0 1])
title('\bf CDF')
% Inverse CDF
yi1 = icdf('ncx2',0.5,nu,delta)
yi2 = ncx2inv(0.5,nu,delta)
xx = 0:0.01:1;
yy = icdf('ncx2',xx,nu,delta);
subplot(2,2,3)
plot(xx,yy,'b-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[0 30.12])
title('\bfInverse CDF')
% Random Numbers
rand('twister',5489)
yr1 = random('ncx2',nu,delta,3)
yr1 = ncx2rnd(nu,delta,3)
yran = random('ncx2',nu,delta,1,500);
subplot(2,2,4)
h1 = histogram(yran)
set(gca,'fontsize',11,'fontweigh','bold')
h1.NumBins = 20;
h1.FaceColor = [1 1 1];
h1.EdgeColor = 'black';
title('\bf Histogram')
saveas(gcf,'NoncentralChi2RV101','epsc')
save('NoncentralChi2RV101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 자유도가 $\nu = 9$이고 비중심모수가 $\delta = 3.3$인 비중심카이제곱확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 500개 비중심카이제곱난수들을 생성하고, 이들의 히스토그램을 그린다. 이

그래프들이 그림 1.5.37에 그려져 있다.

비중심카이제곱확률분포에 관한 MATLAB함수들로는 ncx2pdf, pdf, ncx2cdf, cdf, ncx2inv, icdf, ncx2stat, ncx2rnd, random, randtool 등이 있다. ■

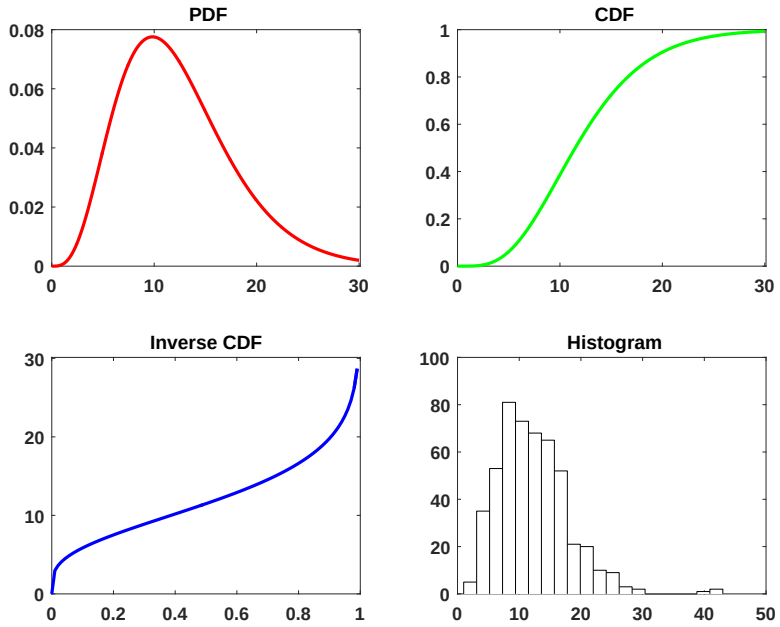


그림 1.5.37. 비중심카이제곱확률변수와 MATLAB

예제 1.5.38 R을 이용해서 비중심카이제곱확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 비중심카이제곱난수를 살펴보기 위해서, 다음 R프로그램 NoncentralChi2RV101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------------
# Filename: NoncentralChi2RV101R.R
# Noncentral Chi-squared Random Variable
# Programmed by CBS
# ------------------------------------------------------------------------------
nAbsissa <- 801
nu <- 9; delta <-3.3
x <- seq(0,30,len=nAbsissa)
yp <- dchisq(x, df=nu, ncp=delta, log=FALSE)
yc <- pchisq(x, df=nu, ncp=delta, lower.tail=TRUE, log.p=FALSE)
ix <- seq(0,1,len=nAbsissa)
yi <- qchisq(ix, df=nu, ncp=delta, lower.tail=TRUE, log.p=FALSE)
nSim <- 500
set.seed(41)
yran <- rchisq(nSim, df=nu, ncp=0)

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
```

```
plot.new()
postscript('NoncentralChi2RV101R.eps')  # Start to save figure
RVdata1 <- data.frame(x,yp,yc)
RVdata2 <- data.frame(ix,yi)
RVdata3 <- data.frame(yran)
plot11 <- ggplot(RVdata1, aes(x,yp)) +
          geom_line(col="red",lwd=1.2) +
          xlab("x") +
          ggtitle("PDF")
plot12 <- ggplot(RVdata1, aes(x,yc)) +
          geom_line(col="green",lwd=1.2) +
          xlab("x") +
          ggtitle("CDF")
plot13 <- ggplot(RVdata2, aes(ix,yi)) +
          geom_line(col="blue",lwd=1.2) +
          xlab("x") +
          ggtitle("Inverse CDF")
plot14 <- ggplot(RVdata3, aes(x=yran)) +
          geom_histogram(bins=20,fill="white",color="black")+
          xlab("x") +
          ggtitle("Histogram")
pushViewport(viewport(layout=grid.layout(2,2)))
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot13, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot14, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()             # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 자유도가 $\nu = 9$이고 비중심모수가 $\delta = 3.3$인 비중심카이제곱확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 500개 비중심카이제곱난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.38에 그려져 있다. ■

1.5.16 비중심 t확률분포

서로 독립인 표준정규확률변수 x와 자유도가 ν인 카이제곱확률변수 y에 대해서 다음과 같은 확률변수를 정의하자.

$$x \doteq \frac{z+\delta}{\sqrt{y/\nu}} \tag{1.5.70}$$

확률변수 x를 자유도가 ν이고 비중심모수가 δ인 비중심 t확률분포를 따른다고 하고, 이를 $t_\nu(\delta)$로 표기한다. 이 확률변수의 평균과 분산은 각각 다음과 같다.

$$E(x) = \delta\sqrt{\frac{\nu}{2}}\frac{\Gamma([\nu-1]/2)}{\Gamma(\nu/2)}, \quad (\nu > 1) \tag{1.5.71}$$

$$Var(x) = \frac{\nu[1+\delta^2]}{\nu-2} - E(x)^2, \quad (\nu > 2) \tag{1.5.72}$$

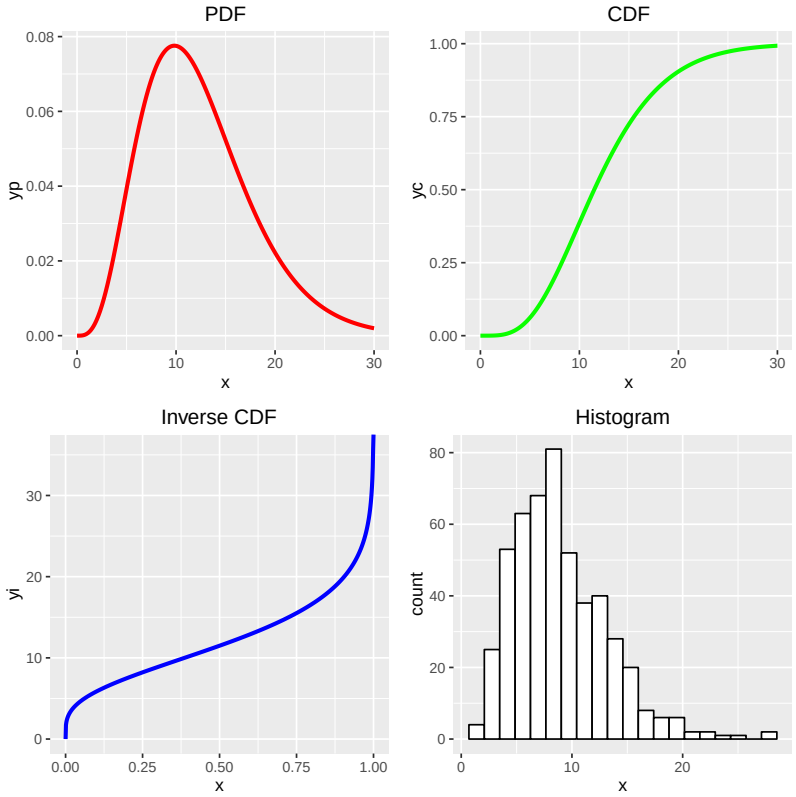


그림 1.5.38. 비중심카이제곱확률변수와 R

예제 1.5.39 MATLAB을 이용해서 비중심 t확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 비중심t난수를 살펴보기 위해서, 다음 MATLAB프로그램 Noncentral_tRV101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: Noncentral_tRV101.m
%  Noncentral t Random Variable
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
nu = 9, delta = 3.3                                % Parameters
% PDF
yp1 = pdf('nct',1,nu,delta)
yp2 = nctpdf(1,nu,delta)
xx = 0:0.05:30;
yy = pdf('nct',xx,nu,delta);
subplot(2,2,1)
plot(xx,yy,'r-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 9.12],'ylim',[0 0.35])
title('\bf PDF')
% CDF
yc1 = cdf('nct',1,nu,delta)
yc2 = nctcdf(1,nu,delta)
xx = 0:0.05:30;
yy = cdf('nct',xx,nu,delta);
subplot(2,2,2)
plot(xx,yy,'g-','linewidth',2)
```

```
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 9.12],'ylim',[0 1])
title('\bf CDF')
% Inverse CDF
yi1 = icdf('nct',0.5,nu,delta)
yi2 = nctinv(0.5,nu,delta)
xx = 0:0.01:1;
yy = icdf('nct',xx,nu,delta);
subplot(2,2,3)
plot(xx,yy,'b-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[0 9.12])
title('\bfInverse CDF')
% Random Numbers
rand('twister',5489)
yr1 = random('nct',nu,delta,3)
yr1 = nctrnd(nu,delta,3)
yran = random('nct',nu,delta,1,500);
subplot(2,2,4)
h1 = histogram(yran)
set(gca,'fontsize',11,'fontweigh','bold')
h1.NumBins = 20;
h1.FaceColor = [1 1 1];
h1.EdgeColor = 'black';
title('\bf Histogram')
saveas(gcf,'Noncentral_tRV101','epsc')
save('Noncentral_tRV101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB 프로그램을 실행하면, 자유도가 $\nu = 9$ 이고 비중심모수가 $\delta = 3.3$ 인 비중심 t 확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 500 개 비중심 t 난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.39에 그려져 있다.

비중심 t 확률분포에 관한 MATLAB 함수들로는 nctpdf, pdf, nctcdf, cdf, nctinv, icdf, nctstat, nctrnd, random, randtool 등이 있다. ■

예제 1.5.40 R을 이용해서 비중심 t 확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 비중심 t 난수를 살펴보기 위해서, 다음 R 프로그램 Noncentral_tRV101R.R 을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: Noncentral_tRV101R.R
# Noncentral t Random Variable
# Programmed by CBS
# ------------------------------------------------------------------------
nAbsissa <- 801
nu <- 9; delta <- 3.3
x <- seq(0,9.12,len=nAbsissa)
yp <- dt(x, df=nu, ncp=delta, log=FALSE)
yc <- pt(x, df=nu, ncp=delta, lower.tail=TRUE, log.p=FALSE)
ix <- seq(0,1,len=nAbsissa)
yi <- qt(ix, df=nu, ncp=delta, lower.tail=TRUE, log.p=FALSE)
```

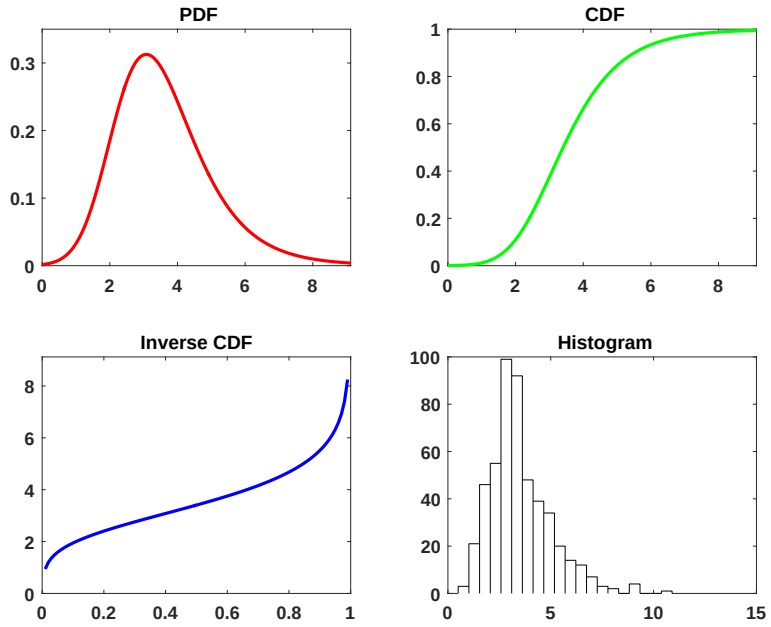


그림 1.5.39. 비중심 t 확률변수와 MATLAB

```
nSim <- 500
set.seed(41)
yran <- rt(nSim, df=nu, ncp=delta)

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('Noncentral_tRV101R.eps')  # Start to save figure
RVdata1 <- data.frame(x,yp,yc)
RVdata2 <- data.frame(ix,yi)
RVdata3 <- data.frame(yran)
plot11 <- ggplot(RVdata1, aes(x,yp)) +
          geom_line(col="red",lwd=1.2) +
          xlab("x") +
          ggtitle("PDF")
plot12 <- ggplot(RVdata1, aes(x,yc)) +
          geom_line(col="green",lwd=1.2) +
          xlab("x") +
          ggtitle("CDF")
plot13 <- ggplot(RVdata2, aes(ix,yi)) +
          geom_line(col="blue",lwd=1.2) +
          xlab("x") +
          ggtitle("Inverse CDF")
plot14 <- ggplot(RVdata3, aes(x=yran)) +
          geom_histogram(bins=20,fill="white",color="black")+
          xlab("x") +
          ggtitle("Histogram")
pushViewport(viewport(layout=grid.layout(2,2)))
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot13, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot14, vp=viewport(layout.pos.row=2, layout.pos.col=2))
```

```
49 dev.off()              # End to save figure
50 # ---------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 자유도가 $\nu = 9$이고 비중심모수가 $\delta = 3.3$인 비중심 t확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 500개 비중심 t난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.40에 그려져 있다. ■

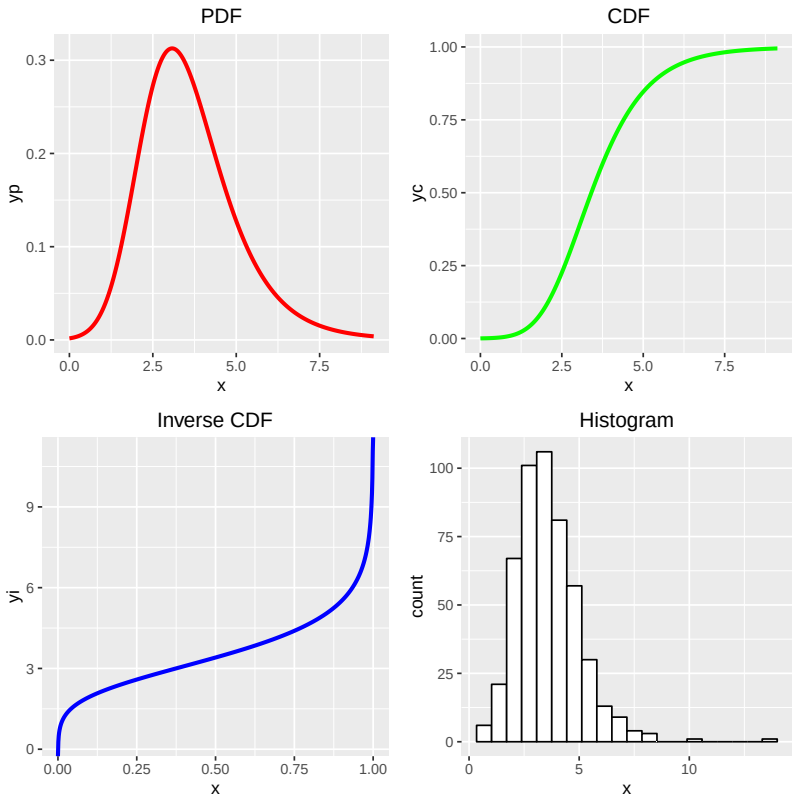


그림 1.5.40. 비중심 t확률변수와 R

1.5.17 비중심 F확률분포

서로 독립인 자유도가 ν_1이고 비중심모수가 δ인 비중심카이제곱확률변수 u와 자유도가 ν_2인 카이제곱확률변수 ν에 대해서 다음과 같은 확률변수를 정의하자.

$$x \doteq \frac{u/\nu_1}{v/\nu 2} \tag{1.5.73}$$

확률변수 x를 자유도가 (ν_1, ν_2)이고 비중심모수가 δ인 F확률변수라고 부르고, 이에 해당하는 확률분포를 $F_{\nu_1,\nu_2}(\delta)$로 표기한다. 이 확률변수의 확률밀도함수는 다음과 같다.

$$f(x) = \sum_{j=0}^{\infty} e^{-\delta/2} \left[\frac{\delta}{2}\right]^j \frac{[\nu_1/\nu_2]^{\nu_1/2+j}}{B(\frac{\nu_1}{2}, \frac{\nu_2}{2}+j))} x^{[\nu_1-2+2j]/2} \left[1+\frac{\nu_1}{\nu_2}x\right]^{-[\nu_1+\nu_2]/2-j} \tag{1.5.74}$$

또한, 평균과 분산은 각각 다음과 같다.

$$E(x) = \frac{[\nu_1+\delta]\nu_2}{\nu_1[\nu_2-2]}, \quad (\nu_2 > 2) \tag{1.5.75}$$

$$Var(x) = \frac{2\{[\nu_1+\delta]^2 + [\nu_1+2\delta][\nu_2-2]\}}{[\nu_2-2]^2[\nu_2-4]} \frac{\nu_2^2}{\nu_1^2}, \quad (\nu_2 > 4) \tag{1.5.76}$$

정의에서 알 수 있듯이, 서로 독립인 비중심카이제곱난수와 중심카이제곱난수를 사용해서, 확률분포 $F_{\nu_1,\nu_2}(\delta)$으로부터 난수를 생성할 수 있다.

예제 1.5.41 MATLAB을 이용해서 비중심 F확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 비중심F난수를 살펴보기 위해서, 다음 MATLAB프로그램 NoncentralF_RV101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: NoncentralF_RV101.m
%  Noncentral F Random Variable
%  Programmed by CBS
%-------------------------------------------------------------------
clear all, close all
nu1 = 9, nu2 = 4, delta = 3.3                    % Parameters
% PDF
yp1 = pdf('ncf',1,nu1,nu2,delta)
yp2 = ncfpdf(1,nu1,nu2,delta)
xx = 0:0.05:30;
yy = pdf('ncf',xx,nu1,nu2,delta);
subplot(2,2,1)
plot(xx,yy,'r-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 14.12],'ylim',[0 0.52])
title('\bf PDF')
% CDF
yc1 = cdf('ncf',1,nu1,nu2,delta)
yc2 = ncfcdf(1,nu1,nu2,delta)
xx = 0:0.05:30;
yy = cdf('ncf',xx,nu1,nu2,delta);
subplot(2,2,2)
plot(xx,yy,'g-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 14.12],'ylim',[0 1])
title('\bf CDF')
% Inverse CDF
yi1 = icdf('ncf',0.5,nu1,nu2,delta)
yi2 = ncfinv(0.5,nu1,nu2,delta)
xx = 0:0.01:1;
yy = icdf('ncf',xx,nu1,nu2,delta);
```

```
subplot(2,2,3)
plot(xx,yy,'b-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[0 14.12])
title('\bfInverse CDF')
% Random Numbers
rand('twister',5489)
yr1 = random('ncf',nu1,nu2,delta,3)
yr1 = ncfrnd(nu1,nu2,delta,3)
yran = random('ncf',nu1,nu2,delta,1,500);
subplot(2,2,4)
h1 = histogram(yran)
set(gca,'fontsize',11,'fontweigh','bold')
h1.NumBins = 20;
h1.FaceColor = [1 1 1];
h1.EdgeColor = 'black';
title('\bf Histogram')
saveas(gcf,'NoncentralF_RV101','epsc')
save('NoncentralF_RV101','yran')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB 프로그램을 실행하면, 자유도가 $(9, 4)$ 이고 비중심모수가 3.3 인 비중심 F 확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 500 개 비중심 F 난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.41에 그려져 있다.

비중심확률분포에 관한 MATLAB 함수들로는 ncfpdf, pdf, ncfcdf, cdf, ncfinv, icdf, ncfstat, ncfrnd, random, randtool 등이 있다. ■

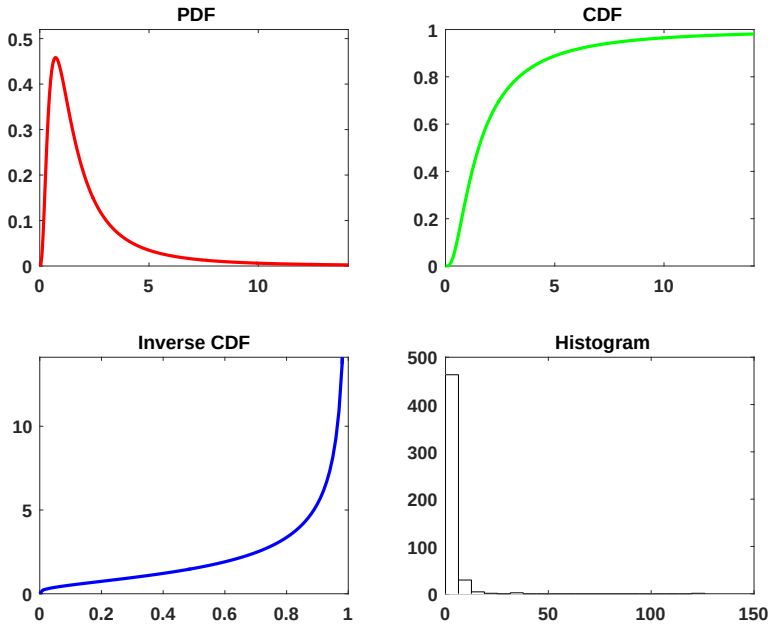


그림 1.5.41. 비중심 F 확률변수와 MATLAB

예제 1.5.42 R을 이용해서 비중심 F 확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 비중심F 난수를 살펴보기 위해서, 다음 R프로그램 NoncentralF_RV101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: NoncentralF_RV101R.R
# Noncentral F Random Variable
# Programmed by CBS
# ------------------------------------------------------------------------
nAbsissa <- 801
nu1 <- 9; nu2 <- 4; delta <- 3.3
x <- seq(0,14.12,len=nAbsissa)
yp <- df(x, df1=nu1, df2=nu2, ncp=delta, log=FALSE)
yc <- pf(x, df1=nu1, df2=nu2, ncp=delta, lower.tail=TRUE, log.p=FALSE)
ix <- seq(0,1,len=nAbsissa)
yi <- qf(ix, df1=nu1, df2=nu2, ncp=delta, lower.tail=TRUE, log.p=FALSE)
nSim <- 500
set.seed(41)
yran <- rf(nSim, df1=nu1,df2=nu2, ncp=delta)

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('NoncentralF_RV101R.eps')  # Start to save figure
RVdata1 <- data.frame(x,yp,yc)
RVdata2 <- data.frame(ix,yi)
RVdata3 <- data.frame(yran)
plot11 <- ggplot(RVdata1, aes(x,yp)) +
          geom_line(col="red",lwd=1.2) +
          xlab("x") +
          ggtitle("PDF")
plot12 <- ggplot(RVdata1, aes(x,yc)) +
          geom_line(col="green",lwd=1.2) +
          xlab("x") +
          ggtitle("CDF")
plot13 <- ggplot(RVdata2, aes(ix,yi)) +
          geom_line(col="blue",lwd=1.2) +
          xlab("x") +
          ggtitle("Inverse CDF")
plot14 <- ggplot(RVdata3, aes(x=yran)) +
          geom_histogram(bins=20,fill="white",color="black")+
          xlab("x") +
          ggtitle("Histogram")
pushViewport(viewport(layout=grid.layout(2,2)))
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot13, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot14, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()             # End to save figure
# ------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 자유도가 $(9, 4)$이고 비중심모수가 3.3인 비중심 F 확률분포의

확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 500 개 비중심 F 난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.42에 그려져 있다. ■

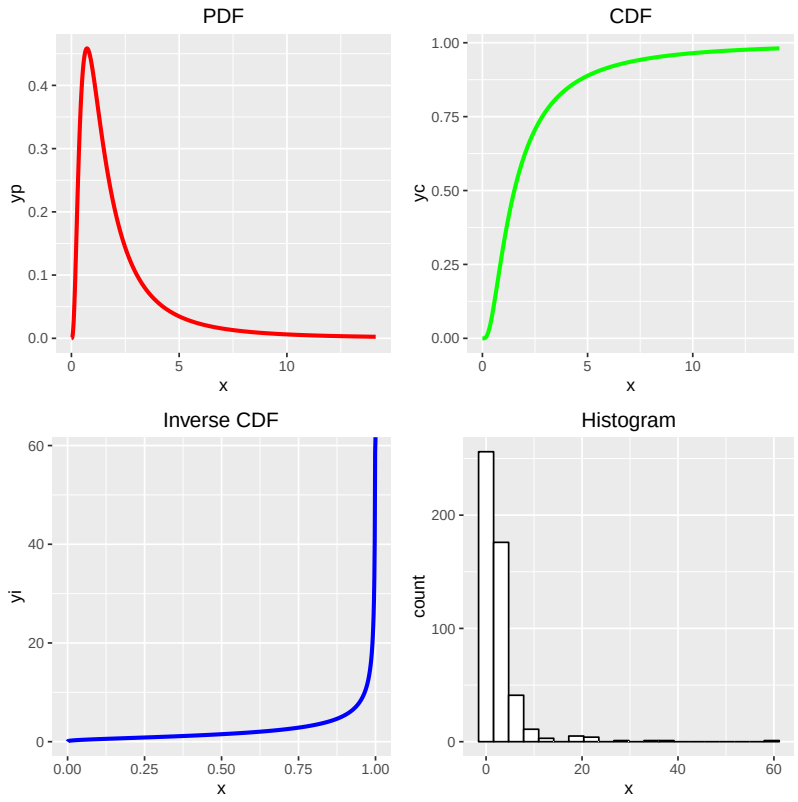


그림 1.5.42. 비중심 F 확률변수와 R

1.5.18 로지스틱확률분포

위치모수가 μ 이고 형태모수가 σ 인 로지스틱확률분포 (logistic probability distribution)의 확률밀도함수는 다음과 같다.

$$f(x) = \frac{\exp(-\frac{x-\mu}{\sigma})}{\sigma[1+\exp(-\frac{x-\mu}{\sigma})]^2} \tag{1.5.77}$$

또한, 누적확률분포함수는 다음과 같다.

$$F(x) = \frac{1}{1+\exp(-\frac{x-\mu}{\sigma})} \tag{1.5.78}$$

로지스틱확률분포함수는 생장곡선으로서 S형이다. 이 로지스틱확률분포의 평균과 분산은 각각 다음과 같다.

$$E(x) = \mu, \quad Var(x) = \frac{\pi^2}{3}\sigma^2\mu^2 \tag{1.5.79}$$

또한, 적률모함수(moment generating function)은 다음과 같다.

$$m_x(t) = e^{\mu t} Beta(1-\sigma t, 1+\sigma t), \quad \left(t \in \left(-\frac{1}{\sigma}, \frac{1}{\sigma}\right)\right) \tag{1.5.80}$$

예제 1.5.43 MATLAB을 이용해서 로지스틱확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 로지스틱난수를 살펴보기 위해서, 다음 MATLAB프로그램 LogisticRV101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: LogisticRV101.m
%  Logistic Random Variable
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
mu = 0, sigma = 1.5                          % Parameters
% PDF
yp1 = pdf('logistic',1,mu,sigma)
xx = -10:0.05:10;
yy = pdf('logistic',xx,mu,sigma);
subplot(2,2,1)
plot(xx,yy,'r-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
title('\bf PDF')
% CDF
yc1 = cdf('logistic',1,mu,sigma)
xx = -10:0.05:10;
yy = cdf('logistic',xx,mu,sigma);
subplot(2,2,2)
plot(xx,yy,'g-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
title('\bf CDF')
% Inverse CDF
yi1 = icdf('logistic',0.5,mu,sigma)
xx = linspace(0,1,201);
yy = icdf('logistic',xx,mu,sigma);
subplot(2,2,3)
plot(xx,yy,'b-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
title('\bfInverse CDF')
% Random Numbers
rand('twister',5489)
yran = random('logistic',mu,sigma,1,500);
subplot(2,2,4)
h1 = histogram(yran)
set(gca,'fontsize',11,'fontweigh','bold')
h1.NumBins = 20;
h1.FaceColor = [1 1 1];
h1.EdgeColor = 'black';
```

```
title('\bf Histogram')
saveas(gcf,'LogisticRV101','epsc')
save('LogisticRV101','yran')
% End of program
% --------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 위치모수가 $\mu = 2$이고 형태모수가 $\sigma = 1.5$인 로지스틱확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 500개 로지스틱난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.43에 그려져 있다.

로지스틱확률분포에 관한 MATLAB함수들로는 pdf, cdf, icdf, mle, fitdist, dfittool, random 등이 있다. ■

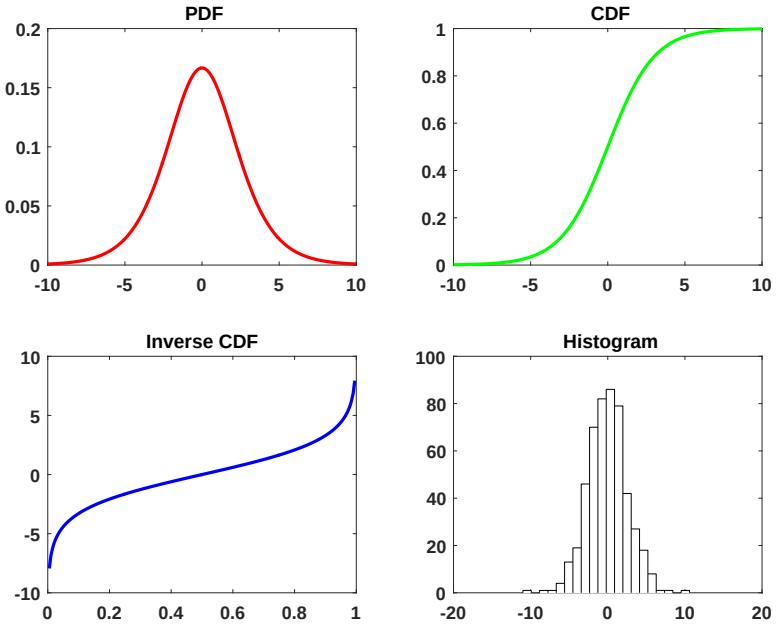


그림 1.5.43. 로지스틱확률변수와 MATLAB

예제 1.5.44 R을 이용해서 로지스틱확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 로지스틱난수를 살펴보기 위해서, 다음 R프로그램 LogisticRV101R.m을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: LogisticRV101R.R
# Logistic Random Variable
# Programmed by CBS
# ------------------------------------------------------------------------
nAbsissa <- 801
x <- seq(-20,20,len=nAbsissa)
yp <- dlogis(x, location=0, scale=1, log=FALSE)   # PDF
```

```
yc <- plogis(x, location=0, scale=1, lower.tail=TRUE, log.p=FALSE)  # CDF
ix <- seq(0,1,len=nAbsissa)
yi <- qlogis(ix, location=0, scale=1,lower.tail=TRUE, log.p=FALSE) # iCDF
nSim <- 500
set.seed(41)
yran <- rlogis(nSim, location=0, scale=1)

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('LogisticRV101R.eps')  # Start to save figure
RVdata1 <- data.frame(x,yp,yc)
RVdata2 <- data.frame(ix,yi)
RVdata3 <- data.frame(yran)
plot11 <- ggplot(RVdata1, aes(x,yp)) +
          geom_line(col="red",lwd=1.2) +
          xlab("x") +
          ggtitle("PDF")
plot12 <- ggplot(RVdata1, aes(x,yc)) +
          geom_line(col="green",lwd=1.2) +
          xlab("x") +
          ggtitle("CDF")
plot13 <- ggplot(RVdata2, aes(ix,yi)) +
          geom_line(col="blue",lwd=1.2) +
          xlab("x") +
          ggtitle("Inverse CDF")
plot14 <- ggplot(RVdata3, aes(x=yran)) +
          geom_histogram(binwidth=0.25,fill="white",color="black")+
          xlab("x") +
          ggtitle("Histogram")
pushViewport(viewport(layout=grid.layout(2,2)))
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot13, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot14, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()            # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 위치모수가 $\mu = 2$이고 형태모수가 $\sigma = 1.5$인 로지스틱확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 500개 로지스틱난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.44에 그려져 있다. ■

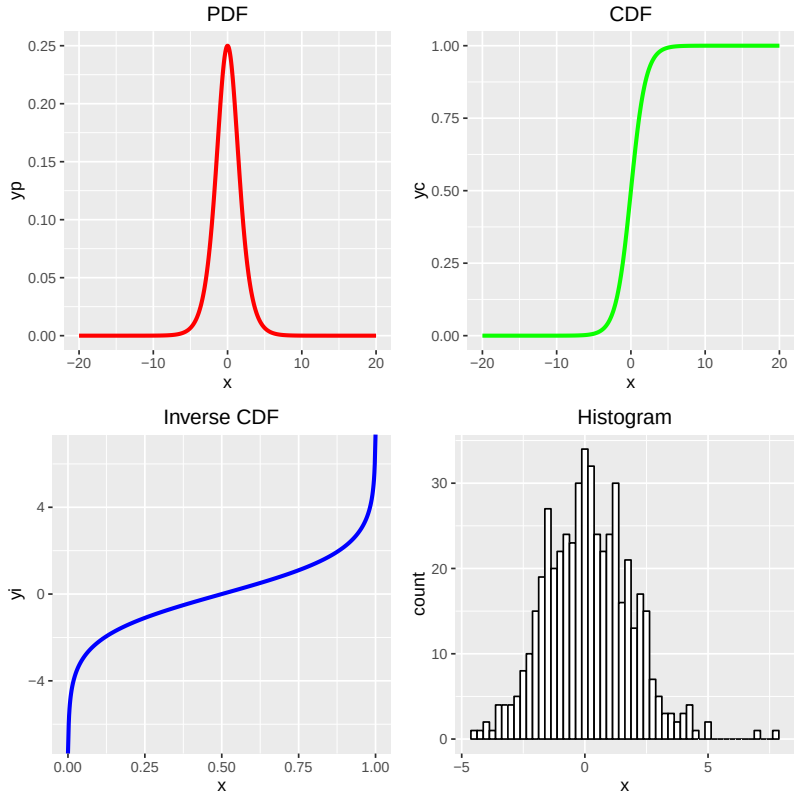


그림 1.5.44. 로지스틱확률변수와 R

1.5.19 역정규확률분포

위치모수가 $\mu(>0)$이고 형태모수가 $\sigma(>0)$인 역정규확률분포(inverse Gaussian probability distribution)의 확률밀도함수는 다음과 같다.

$$f(x)=\sqrt{\frac{\lambda}{2\pi x^3}}\exp\left(-\frac{\lambda}{2\mu^2 x}[x-\mu]^2\right)1_{(0,\infty)}(x) \tag{1.5.81}$$

또한, 누적확률분포함수는 다음과 같다.

$$F(x)=\left[N\left(\sqrt{\frac{\lambda}{x}}\left[\frac{x}{\mu}-1\right]\right)+\exp\left(\frac{2\lambda}{\mu}\right)N\left(-\sqrt{\frac{\lambda}{x}}\left[\frac{x}{\mu}+1\right]\right)\right]1_{(0,\infty)}(x) \tag{1.5.82}$$

이 역정규확률분포는 Brown운동에서 정해진 레벨에 도달하는데 걸리는 시간의 확률분포이다. 이 역정규확률분포의 평균과 분산은 각각 다음과 같다.

$$E(x)=\mu,\quad Var(x)=\frac{\mu^3}{\lambda} \tag{1.5.83}$$

또한, 적률모함수(moment generating function)는 다음과 같다.

$$m_x(t) = \exp\left(\frac{\lambda}{\mu}\right)\left[1 - \sqrt{1 - \frac{2\mu^2 t}{\lambda}}\right] \tag{1.5.84}$$

예제 1.5.45 MATLAB을 이용해서 역정규확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 역정규난수를 살펴보기 위해서, 다음 MATLAB프로그램 InverseGaussRV1.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: InverseGaussRV101.m
%  Inverse Gaussian Random Variable
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
mu = 1, lambda = 1                          % Parameters
% PDF
yp1 = pdf('inversegaussian',1,mu,lambda)
xx = 0:0.02:6;
yy = pdf('inversegaussian',xx,mu,lambda);
subplot(2,2,1)
plot(xx,yy,'r-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
title('\bf PDF')
% CDF
yc1 = cdf('inversegaussian',1,mu,lambda)
xx = -0:0.02:6;
yy = cdf('inversegaussian',xx,mu,lambda);
subplot(2,2,2)
plot(xx,yy,'g-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
title('\bf CDF')
% Inverse CDF
yi1 = icdf('inversegaussian',0.5,mu,lambda)
xx = linspace(0,1,201);
yy = icdf('inversegaussian',xx,mu,lambda);
subplot(2,2,3)
plot(xx,yy,'b-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
title('\bfInverse CDF')
% Random Numbers
rand('twister',5489)
yran = random('inversegaussian',mu,lambda,1,500);
subplot(2,2,4)
h1 = histogram(yran)
set(gca,'fontsize',11,'fontweigh','bold')
h1.NumBins = 20;
h1.FaceColor = [1 1 1];
h1.EdgeColor = 'black';
set(gca,'fontsize',11,'fontweigh','bold')
title('\bf Histogram')
saveas(gcf,'InverseGaussRV101','epsc')
save('InverseGaussRV101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 위치모수가 $\mu = 1$이고 형태모수가 $\lambda = 1$인 역정규확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 500개 역정규난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.45에 그려져 있다.

역정규확률분포에 관한 MATLAB함수들로는 pdf, cdf, icdf, mle, fitdist, dfittool, random 등이 있다. ■

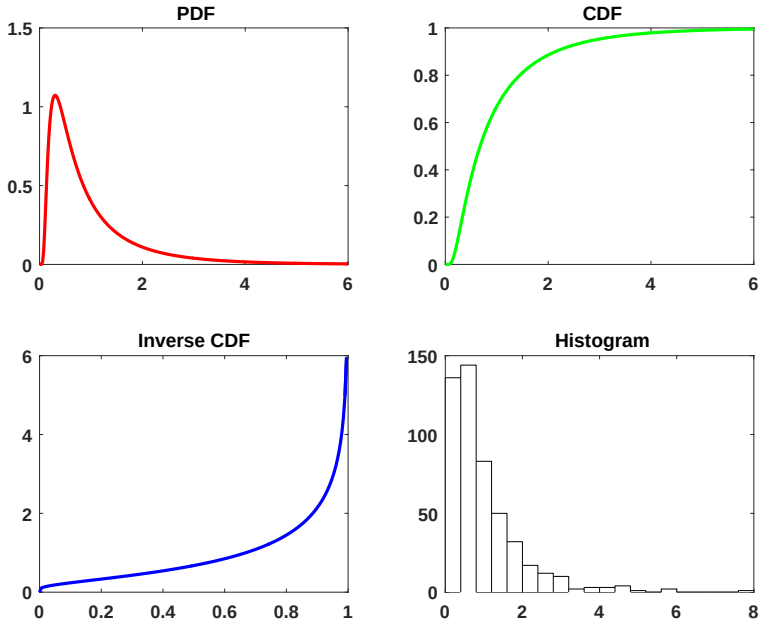


그림 1.5.45. 역정규확률변수와 MATLAB

예제 1.5.46 R을 이용해서 역정규확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 역정규난수를 살펴보기 위해서, 다음 R프로그램 InverseGaussRV1R.R을 실행해 보자.

```
# ------------------------------------------------------------------------------
# Filename InverseGaussRV101R.R
# Inverse Gaussian Random Variable
# Programmed by CBS
# ------------------------------------------------------------------------------
# install.packages("statmod")
library(statmod)
mu <- 1; lambda <- 1
nAbsissa <- 801
x <- seq(0,6,len=nAbsissa)
yp <- dinvgauss(x, 1, 1)    # PDF
yc <- pinvgauss(x, 1, 1)    # CDF
ix <- seq(0,1,len=nAbsissa)
yi <- qinvgauss(ix, 1, 1)  #Inverse CDF
nSim <- 500
set.seed(41)
```

```
yran <- rinvgauss(nSim, 1, 1)

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('InverseGaussRV101R.eps')  # Start to save figure
RVdata1 <- data.frame(x,yp,yc)
RVdata2 <- data.frame(ix,yi)
RVdata3 <- data.frame(yran)
plot11 <- ggplot(RVdata1, aes(x,yp)) +
          geom_line(col="red",lwd=1.2) +
          xlab("x") +
          ggtitle("PDF")
plot12 <- ggplot(RVdata1, aes(x,yc)) +
          geom_line(col="green",lwd=1.2) +
          xlab("x") +
          ggtitle("CDF")
plot13 <- ggplot(RVdata2, aes(ix,yi)) +
          geom_line(col="blue",lwd=1.2) +
          xlab("x") +
          ggtitle("Inverse CDF")
plot14 <- ggplot(RVdata3, aes(x=yran)) +
          geom_histogram(bins=20,fill="white",color="black")+
          xlab("x") +
          ggtitle("Histogram")
pushViewport(viewport(layout=grid.layout(2,2)))
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot13, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot14, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()            # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 위치모수가 $\mu = 1$이고 형태모수가 $\lambda = 1$인 역정규확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 500개 역정규난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.46에 그려져 있다. ■

1.5.20 역감마확률분포

형태모수가 $\alpha(>0) = 3$이고 척도모수가 $\beta(>0) = 1$인 역감마확률분포(inverse gamma probability distribution)의 확률밀도함수는 다음과 같다.

$$f(x;\alpha,\beta) = \frac{\beta^{\alpha}}{\Gamma(\alpha)} x^{-\alpha-1} \exp\left(-\frac{\beta}{x}\right) 1_{(0,\infty)}(x) \tag{1.5.85}$$

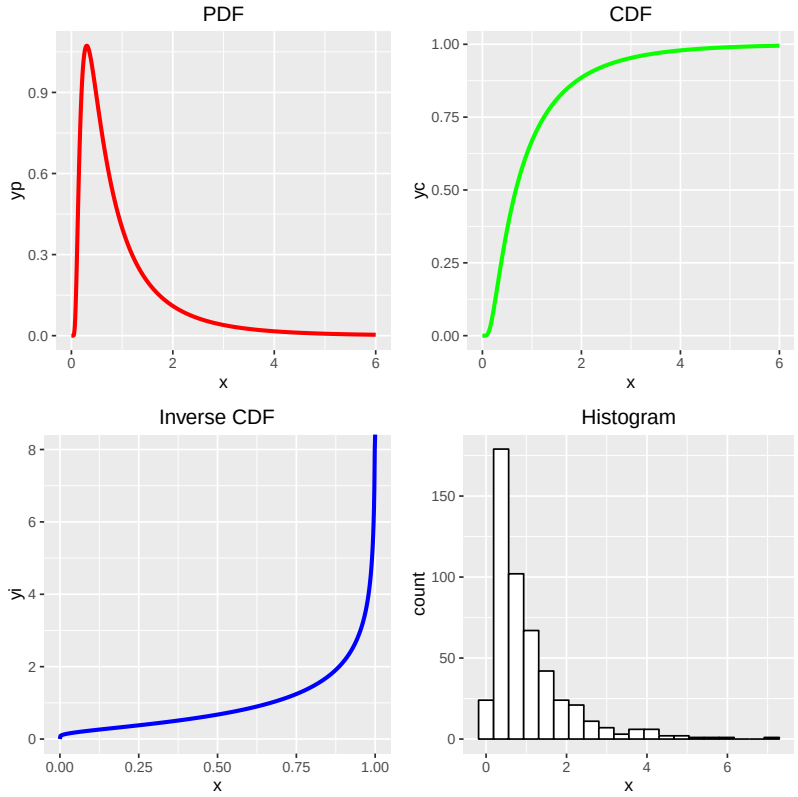


그림 1.5.46. 역정규확률변수와 R

또한, 누적확률분포함수는 다음과 같다.

$$F(x;\alpha,\beta)=\frac{\Gamma\left(\alpha,\frac{\beta}{x}\right)}{\Gamma(\alpha)}\,1_{(0,\infty)}(x) \tag{1.5.86}$$

이 역감마확률분포의 평균과 분산은 각각 다음과 같다.

$$E(x)=\frac{\beta}{\alpha-1},\quad(\alpha>1) \tag{1.5.87}$$

$$Var(x)=\frac{\beta^2}{[\alpha-1]^2[\alpha-2]},\quad(\alpha>2) \tag{1.5.88}$$

예제 1.5.47 MATLAB을 이용해서 역감마확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 역감마난수를 살펴보기 위해서, 다음 MATLAB프로그램 InverseGammaRV1.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: InverseGammaRV101.m
%  Inverse Gamma Random Variable
%  Programmed by CBS
%------------------------------------------------------------------
```

```
clear all, close all
alpha = 3, beta = 1                          % Parameters
% PDF
xx = 0:0.02:6;
yy = beta^alpha/gamma(alpha)*xx.^(-alpha-1).*exp(-beta./xx);
subplot(2,2,1)
plot(xx,yy,'r-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
title('\bf PDF')
% CDF
xx = -0:0.02:6;
yy = 1 - gamcdf(1./xx,alpha,1/beta);;
subplot(2,2,2)
plot(xx,yy,'g-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
title('\bf CDF')
% Inverse CDF
xx = linspace(0,1,201);
yy = 1./gaminv(1 - xx,alpha,beta);
subplot(2,2,3)
plot(xx,yy,'b-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
title('\bfInverse CDF')
% Random Numbers
rand('twister',5489)
yran = 1./gamrnd(alpha,beta,500,1);
subplot(2,2,4)
h1 = histogram(yran)
set(gca,'fontsize',11,'fontweigh','bold')
h1.NumBins = 20;
h1.FaceColor = [1 1 1];
h1.EdgeColor = 'black';
set(gca,'fontsize',11,'fontweigh','bold')
title('\bf Histogram')
saveas(gcf,'InverseGammaRV101','epsc')
save('InverseGammaRV101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 형태모수가 $\alpha = 3$이고 척도모수가 $\beta(> 0) = 1$인 역감마확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 500개 역감마난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.47에 그려져 있다.

역감마확률분포에 관한 MATLAB함수들로는 pdf, cdf, icdf, mle, fitdist, dfittool, random 등이 있다. ■

예제 1.5.48 R을 이용해서 역감마확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 역감마난수를 살펴보기 위해서, 다음 R프로그램 InverseGammaRV1R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
```

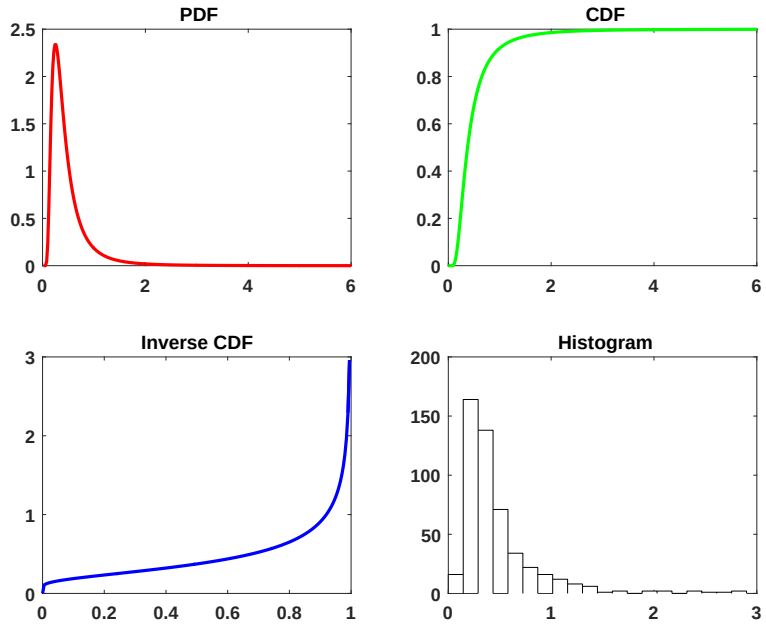


그림 1.5.47. 역감마확률변수와 MATLAB

```
# Filename InverseGammaRV101R.R
# Inverse Gamma Random Variable
# Programmed by CBS
# ------------------------------------------------------------------------------
# install.packages("pscl")
library(pscl)
alpha <- 3; betaa <- 1
nAbsissa <- 801
x <- seq(0,6,len=nAbsissa)
yp <- densigamma(x,alpha,beta)                # PDF
yc <- pigamma(q,alpha,beta)                   # CDF
ix <- seq(0,1,len=nAbsissa)
yi <- qigamma(p,alpha,beta)                   # iCDF
nSim <- 500
set.seed(41)
yran <- rigamma(n,alpha,beta)                 # random number


# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('InverseGammaRV101R.eps')  # Start to save figure
RVdata1 <- data.frame(x,yp,yc)
RVdata2 <- data.frame(ix,yi)
RVdata3 <- data.frame(yran)
plot11 <- ggplot(RVdata1, aes(x,yp)) +
          geom_line(col="red",lwd=1.2) +
          xlab("x") +
          ggtitle("PDF")
plot12 <- ggplot(RVdata1, aes(x,yc)) +
          geom_line(col="green",lwd=1.2) +
          xlab("x") +
```

```
          ggtitle("CDF")
plot13 <- ggplot(RVdata2, aes(ix,yi)) +
          geom_line(col="blue",lwd=1.2) +
          xlab("x") +
          ggtitle("Inverse CDF")
plot14 <- ggplot(RVdata3, aes(x=yran)) +
          geom_histogram(bins=20,fill="white",color="black")+
          xlab("x") +
          ggtitle("Histogram")
pushViewport(viewport(layout=grid.layout(2,2)))
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot13, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot14, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()             # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 형태모수가 $\alpha =$이고 척도모수가 $\beta(>0)$인 역감마확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 500개 역감마난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.47에 그려져 있다. ■

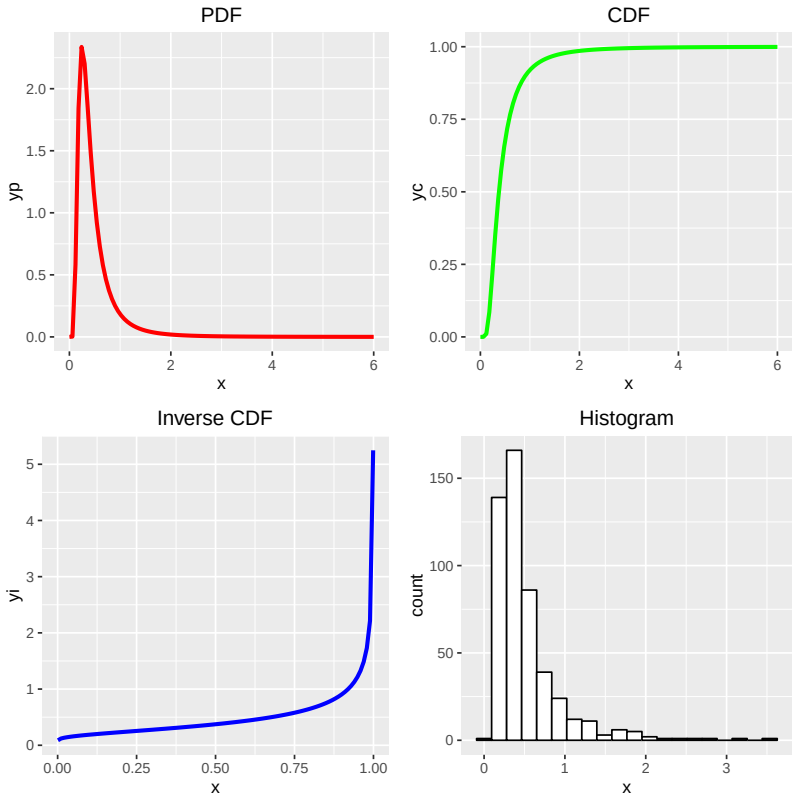


그림 1.5.48. 역감마확률변수와 R

1.5.21 Cauchy확률분포

서로 독립인 표준정규확률변수들 v_1과 v_2에 대해서 다음과 같은 확률변수를 정의하자.

$$x \doteq \frac{v_1}{v_2} \tag{1.5.89}$$

확률변수 x를 Cauchy확률변수라고 부른다. 이 확률변수의 확률밀도함수는 다음과 같다.

$$f(x) = \frac{1}{\pi}\frac{1}{1+x^2} \tag{1.5.90}$$

또한, 누적확률분포함수와 그 역함수는 각각 다음과 같다.

$$F(x) = \frac{1}{\pi}\tan^{-1} x + \frac{1}{2} \tag{1.5.91}$$

$$F^{-1}(x) = \tan\left(\pi x - \frac{\pi}{2}\right) \tag{1.5.92}$$

식 (1.5.92)을 사용해서 Cauchy난수를 생성할 수 있다.

예제 1.5.49 MATLAB을 이용해서 Cauchy확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 Cauchy난수를 살펴보기 위해서, 다음 MATLAB프로그램 CauchyRV101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: CauchyRV101.m
%  Cauchy Random Variable
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
% PDF
xx = linspace(-20,20,41);
f = @(x) 1/pi./(1+x.^2)
yy = f(xx);
subplot(2,2,1)
plot(xx,yy,'r-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
title('\bf PDF')
% CDF
F = @(x) 1/pi*atan(x) + 1/2
yy = F(xx);
subplot(2,2,2)
plot(xx,yy,'g-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
title('\bf CDF')
% Inverse CDF
InF = @(x) tan(pi*x - pi/2)
xx = 0:0.01:1;
yy = InF(xx);
```

```
subplot(2,2,3)
plot(xx,yy,'b-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[-20 20])
title('\bfInverse CDF')
% Random Numbers
n = 10000                              % Number of Random Numbers
rand('twister',5489)
v1 = randn(1,n);
v2 = randn(1,n);
yran = v1./v2;
subplot(2,2,4)
xx = -20:1:20;
hbar = hist(yran,xx);
bar(xx,hbar,'w')
set(gca,'fontsize',11,'fontweigh','bold','xlim',[-20 20])
title('\bf Histogram')
saveas(gcf,'CauchyRV101','epsc')
save('CauchyRV101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, Cauchy확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 10000개 Cauchy난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.49에 그려져 있다.

Cauchy확률분포에 관한 MATLAB함수들로는 MATLAB Statistics Toolbox에 들어 있는 stat::cauchyPDF, stat::cauchyCDF, stat::cauchyQuantile, stat::cauchyRandom 등이 있다.

■

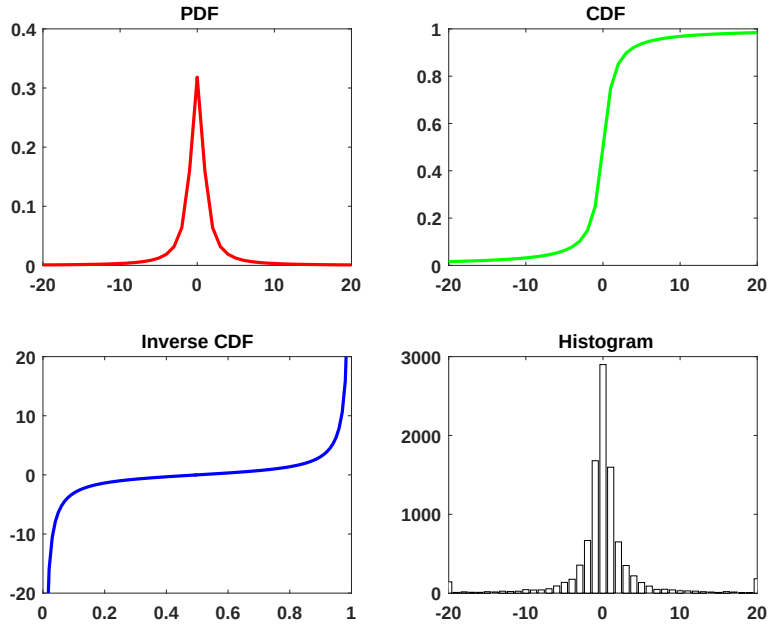


그림 1.5.49. Cauchy확률변수

예제 1.5.50 R을 이용해서 Cauchy확률변수의 확률밀도함수, 누적확률분포함수, 역누적확률분포함수 그리고 Cauchy난수를 살펴보기 위해서, 다음 R프로그램 CauchyRV101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: CauchyRV101R.R
# Cauchy Random Variable
# Programmed by CBS
# ------------------------------------------------------------------------
nAbsissa <- 801
x <- seq(-20,20,len=nAbsissa)
yp <- dcauchy(x, location=0, scale=1, log=FALSE)   # PDF
yc <- pcauchy(x, location=0, scale=1, lower.tail=TRUE, log.p=FALSE)  # CDF
ix <- seq(0,1,len=nAbsissa)
yi <- qcauchy(ix, location=0, scale=1,lower.tail=TRUE, log.p=FALSE) # iCDF
nSim <- 10000
set.seed(41)
yran <- rcauchy(nSim, location=0, scale=1)

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('CauchyRV101R.eps')  # Start to save figure
RVdata1 <- data.frame(x,yp,yc)
RVdata2 <- data.frame(ix,yi)
RVdata3 <- data.frame(yran)
plot11 <- ggplot(RVdata1, aes(x,yp)) +
          geom_line(col="red",lwd=1.2) +
          xlab("x") +
          ggtitle("PDF")
plot12 <- ggplot(RVdata1, aes(x,yc)) +
          geom_line(col="dark green",lwd=1.2) +
          xlab("x") + xlim(-20,20) +
          ggtitle("CDF")
plot13 <- ggplot(RVdata2, aes(ix,yi)) +
          geom_line(col="blue",lwd=1.2) +
          xlab("x") + ylim(-20,20) +
          ggtitle("Inverse CDF")
plot14 <- ggplot(RVdata3, aes(x=yran)) +
          geom_histogram(binwidth=0.25,fill="white",color="black")+
          xlab("x") + xlim(-20,20) +
          ggtitle("Histogram")
pushViewport(viewport(layout=grid.layout(2,2)))
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot13, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot14, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()            # End to save figure
# ------------------------------------------------------------------------
```

이 R프로그램을 실행하면, Cauchy확률분포의 확률밀도함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 10000개 Cauchy난수들을 생성하고,

이들의 히스토그램을 그린다. 이 그래프들이 그림 1.5.50에 그려져 있다. ■

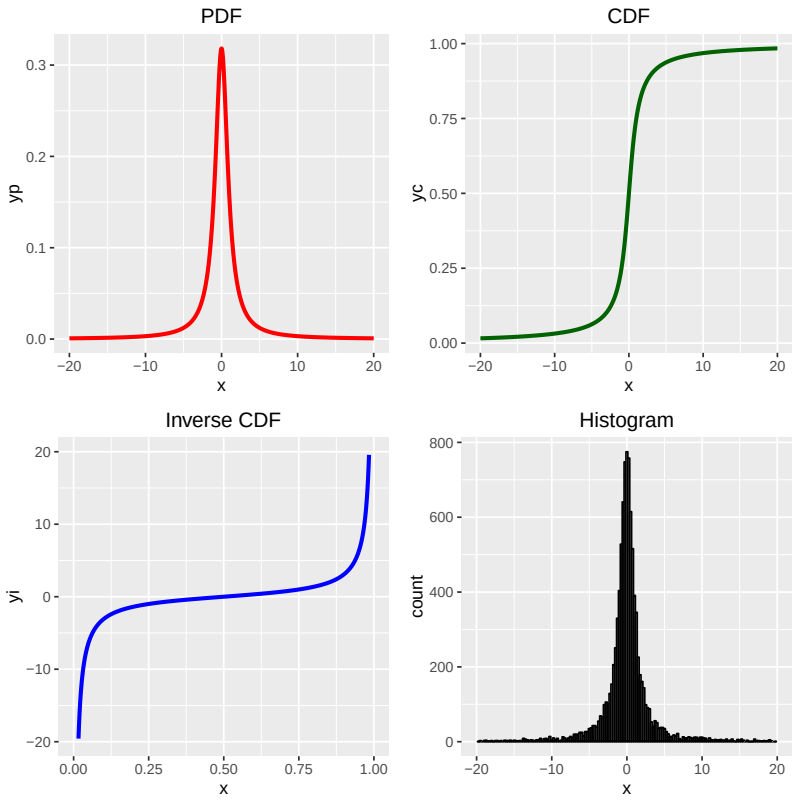


그림 1.5.50. Cauchy확률변수

1.5.22 커널평활확률분포

주어진 관찰값들에 적합한 확률밀도함수, 누적확률분포함수 또는 역누적확률분포함수를 추정할 때 가장 널리 사용되는 비모수적(nonparametric) 방법은 커널평활법(kernel smoothing method)이다. 이 소절에서는 이 방법에 대해서 살펴보자. 어떤 점 x에서 확률밀도함수 f의 값 $f(x)$를 비모수적으로 추정하는 문제를 생각해보자. 충분히 작은 $h(>0)$에 대해서 다음 식을 만족하는 $\hat{\theta}$는 $f(x)$와 아주 가까울 것이다.

$$\hat{\theta} = \arg\min_{\theta} \int_{x-h}^{x+h} [f(y) - \theta]^2 dy \tag{1.5.93}$$

다음 식들이 성립한다.

$$\hat{\theta} = \frac{1}{h} \int_{x-h}^{x+h} [f(y)] dy = \frac{F(x+h) - F(x-h)}{h} \tag{1.5.94}$$

여기서 F는 누적확률분포함수이다. 주어진 관찰값들 $x_1, x_2, \cdots, x_n$에 대해서 누적확률분포함수 F를 다음과 같은 경험확률분포함수로 추정할 수 있다.

$$F_n(x) = \frac{1}{n}\sum_{j=1}^{n} 1(x_j \le x) \tag{1.5.95}$$

따라서, 식 (1.5.94)의 우변을 다음과 같이 추정할 수 있다.

$$\frac{1}{h}[F_n(x+h) - F_n(x-h)] = \frac{1}{n}\sum_{j=1}^{n}\frac{1}{h}1\left(-1 < \frac{x-x_j}{h} \le 1\right) \tag{1.5.96}$$

함수 K를 지지대가 $(-1, 1]$인 일양확률밀도함수라고 하면, 식 (1.5.96)을 다음과 같이 쓸 수 있다.

$$\hat{f}(x) \doteq \frac{1}{n}\sum_{j=1}^{n}\frac{1}{h}K\left(\frac{x-x_j}{h}\right) = \frac{1}{n}\sum_{j=1}^{n} K_h\left(x - x_j\right) \tag{1.5.97}$$

여기서 $K_h(\cdot) \doteq K(\cdot/h)/h$이다.

식 (1.5.97)에서 K를 일양확률밀도함수로 한정할 필요는 없으며, 원점 O에 관해 좌우대칭인 임의의 확률밀도함수로 확장할 수 있다. 이러한 K를 커널 그리고 h를 대역폭(bandwidth)이라 한다. 이 $\hat{f}(x)$를 다음과 같이 쓸 수 있다.

$$\hat{f}(x) = \int_{-\infty}^{\infty} K_h(x-y)dF_n(y) \tag{1.5.98}$$

이 $\hat{f}(x)$를 커널평활확률밀도함수(kernel smoothing probability density function)라 하고 이에 해당하는 누적확률분포함수를 커널평활확률분포함수라 부른다. 대역폭 h가 작을수록 커널평활확률분포함수는 경험확률분포에 가까운 것을 알 수 있다. 그러나, 대역폭 h가 너무 작으면, 커널평활확률분포함수는 매끄러움을 잃어버린다. 만약 n이 커짐에 따라 대역폭 h를 차수 $O(n^{-1/5})$로 에 접근시켜가면, 커널평활확률밀도함수는 차수 $n^{-4/5}$의 속도로 확률밀도함수 $f(x)$에 수렴한다.

예제 1.5.51 MATLAB을 이용해서 커널평활확률밀도함수, 커널평활확률분포함수 그리고 커널평활역확률분포함수를 구하기 위해서 다음 MATLAB프로그램 NonparametricRV101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: NonparametricRV101.m
%  Nonparametric Density Estimation
%  Programmed by CBS
```

```
% ------------------------------------------------------------------
clear all, close all, clf
% Generate Ransom Samples
rand('twister',5489)
y1 = exprnd(2,100,1)+2;
y2 = -exprnd(2,200,1)-1;
yRan = [ y1 ; y2 ];
mu = mean(yRan); sigma = std(yRan);
% Histogram
xleft_Hist = mu-4*sigma;
xright_Hist= mu+4*sigma;
incr_Hist = sigma/6;
xx = ( (xleft_Hist+(incr_Hist/2)) : incr_Hist : xright_Hist )';
% Empirical PDF using Kernel-Smoothing
[Nbar, xcenter] = hist(yRan, xx);
Nheight = Nbar / length(yRan) / (xcenter(2)-xcenter(1));
[ffyRan, fxiyRan] = ksdensity(yRan,xx,'kernel','triangle');
subplot(2,2,1)
hold on
bar(xcenter,Nheight,0.95,'w');                  % Histogram
plot(fxiyRan, ffyRan,'r','LineWidth',[2.5]);    % Empirical PDF
set(gca,'fontsize',11,'fontweigh','bold','xlim',[-15 15])
xlabel('\bf x','fontsize',12), ylabel('\bf PDF','fontsize',12)
hold off;
% Empirical CDF using Kernel-Smoothing
CumNheight = cumsum(Nheight);
CumNheight = CumNheight/CumNheight(end);
[cfyRan, cfxiyRan] = ksdensity(yRan,xx,'function','cdf');
subplot(2,2,2)
hold on
bar(xcenter,CumNheight,0.95,'w');                  % Histogram
plot(cfxiyRan, cfyRan,'g','LineWidth',[2.5]);    % Empirical PDF
set(gca,'fontsize',11,'fontweigh','bold','xlim',[-15 15])
xlabel('\bf x','fontsize',12), ylabel('\bf CDF','fontsize',12)
hold off;
% Empirical Inverse CDF using Kernel-Smoothing
prob = linspace(0.01,0.99,99);
[ifyRan, ifxiyRan] = ksdensity(yRan,prob,'function','icdf');
subplot(2,2,3)
plot(ifxiyRan, ifyRan,'b','LineWidth',[2.5]);    % Empirical PDF
set(gca,'fontsize',11,'fontweigh','bold', ...
    'xlim',[0 1],'ylim',[-15 15])
xlabel('\bf x','fontsize',12)
ylabel('\bf Inverse CDF','fontsize',12)
% Censoring PDF using Kernel-Smoothing
yCensor = ( yRan < mu-1*sigma)|(yRan > mu+1*sigma);
[ffyCensor, fxiyCensor] = ksdensity(yRan,xx,'censoring',yCensor);
subplot(2,2,4)
hold on
bar(xcenter,Nheight,0.95,'w');                      % Histogram
plot(fxiyCensor, ffyCensor,'m','LineWidth',[2.5]); % Empirical PDF
set(gca,'fontsize',11,'fontweigh','bold','xlim',[-15 15])
xlabel('\bf x','fontsize',12)
ylabel('\bf Censoring PDF','fontsize',12)
hold off;
saveas(gcf,'NonparametricRV101','epsc')
save('NonparametricRV101','yRan')
% End of program
% ------------------------------------------------------------------
```

MATLAB함수 ksdensity를 사용하면, 주어진 관찰값들에 적합한 커널평활확률밀도함수, 커널평활확률분포함수 그리고 커널평활역확률분포함수를 추정할 수 있다.
이 MATLAB프로그램의 첫 번째 ksdensity명령문은 다음과 같다.

```
>> [ffyran, fxiyRan] = ksdensity(yRan,xx,'kernel','triangle')
```

여기서 입력변수 yRan은 관찰값들의 벡터이고, xx는 해당 확률변수의 지지대(support)이다. 옵션 kernel은 커널함수를 지정하기 위한 것으로, 선택할 수 있는 커널함수들로는 디폴트인 normal, box, triangle, epanechnikov 등이 있다. 출력변수 ffyran은 커널평활확률밀도함수값들의 벡터이고 출력변수 fxiRan은 이에 해당하는 값들의 벡터이다. 이 MATLAB프로그램을 실행하면, 그림 1.5.51의 좌측상단에 이 커널평활확률밀도함수가 그려진다.

두 번째 ksdensity명령문은 다음과 같다.

```
>> [cfyran, cfxiyRan] = ksdensity(yRan,xx,'function','cdf')
```

여기서 입력변수 yRan은 관찰값들의 벡터이고, xx는 해당 확률변수의 지지대이다. 옵션 function에는 확률밀도함수를 지정하는 pdf, 누적확률분포함수를 지정하는 cdf, 역누적확률분포함수를 지정하는 icdf 등을 사용할 수 있다. 이 옵션의 디폴트는 pdf이다. 출력변수 cfyran은 커널평활확률분포함수값들의 벡터이고 출력변수 cfxiyRan은 이에 해당하는 값들의 벡터이다. 이 MATLAB프로그램을 실행하면, 그림 1.5.51의 우측상단에 이 커널평활확률분포함수가 그려진다.

세 번째 ksdensity명령문은 다음과 같다.

```
>> [ifyran, ifxiyRan] = ksdensity(yRan,prob,'function','icdf')
```

여기서 입력변수 yRan은 관찰값들의 벡터이고, prob은 해당 확률밀도함수이다. 출력변수 ifyran은 커널평활역확률분포함수값들의 벡터이고 출력변수 ifxiyRan은 이에 해당하는 값들의 벡터이다. 이 MATLAB프로그램을 실행하면, 그림 1.5.51의 좌측하단에 이 커널평활역확률분포함수가 그려진다.

MATLAB함수 ksdensity를 사용하면 센서된(censored) 데이터세트에 관한 커널평활확률분포에 관한 추정을 할 수 있다. 이 MATLAB프로그램에서는 데이터세트를 센서된 데이터세트로 만들기 위해서 다음 명령문을 사용하였다.

```
>> yCensor = (yRan < mu-1*Sigma)|(yRan > mu+1*Sigma)
```

이 명령문은 확률변수 yRan의 값이 $\mu-\sigma$보다 작거나 $\mu+\sigma$보다 큰 관찰점에는 확률변수 yCensor에 1을 할당하고 그렇지 않으면 0 을 할당한다. 네 번째 ksdensity명령문은 다음과 같다.

```
>> [ffyCensor,fixyCensor] = ksdensity(yRan,xx,'censoring',yCensor)
```

여기서 입력변수 yRan은 관찰값들의 벡터이고, xx는 해당 확률변수의 지지대이다. 옵션 censoring은 센서된 데이터세트에 관한 커널평활확률분포에 관한 추정을 하겠다는 것이고, yCensor는 원래 데이터세트에서 센서될 관찰점들을 나타내는 벡터이다. 이 MATLAB프로그램을 실행하면, 그림 1.5.51의 우측하단에 이 센서된 커널평활확률밀도함수가 그려진다.

커널평활확률분포에 관한 MATLAB함수들로는 ksdensity, fitdist, dfittool 등이 있다. ■

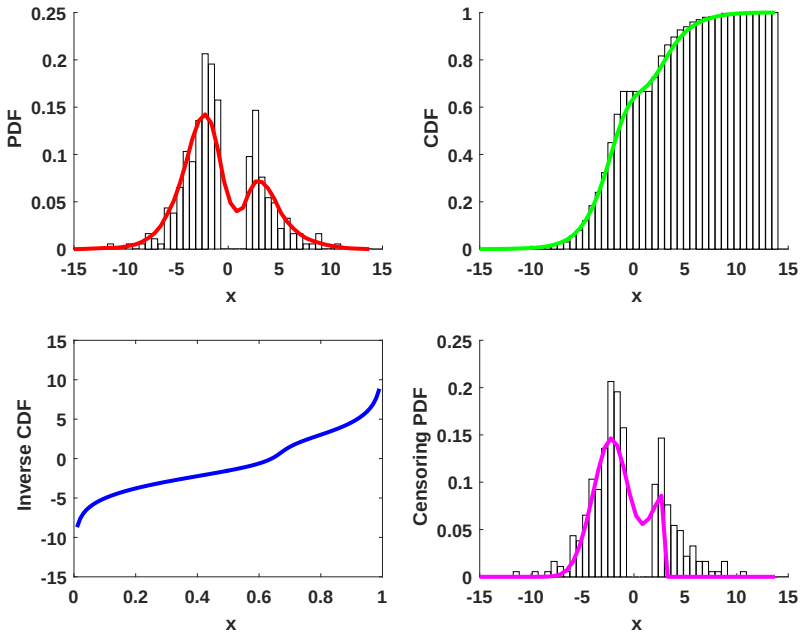


그림 1.5.51. 커널평활확률분포

R을 바탕으로 하는 커널평활확률분포를 다루는 패키지는 다양하다. 예를 들어, GenKern, kerdiest, KernSmooth, ks, np, plugdensity, stat, sm 등이 있다. 또한, R그래픽패키지인 ggplot2에서도 geom_density()나 geom_line()을 해서 커널평활확률밀도함수를 그릴 수 있다.

커널평활확률분포를 다양하게 사용할 수 있다. 다음과 같은 예제들을 살펴보자.

예제 1.5.52 커널평활확률밀도함수를 사용한 비모수적 회귀추정에 관해 살펴보자. 시점 t에서 어떤 금융상품의 수익률을 y_t라 하고 시장포트폴리오의 수익률을 x_t라 한다. 만약 $\{(x_t, y_t) \mid t=1,2,\cdots,n\}$이 서로 독립이며 2변량 정규확률분포를 따른다면, 다음과 같은

선형회귀모형을 사용할 수 있다.

$$y_t = \alpha + \beta x_t + \epsilon_t, \quad (t = 1, 2, \cdots, n) \tag{1}$$

여기서 α와 β는 상수들이고, 오차항들 $\{\epsilon_t\}$는 서로 독립이며 평균이 0인 정규확률분포를 따른다. 그러나, x와 y가 반드시 식 (1)과 같은 선형관계에 있다고는 할 수 없으므로, 다음과 같은 식이 성립한다고 가정하자.

$$y_t = g(x_t) + \epsilon_t, \quad (t = 1, 2, \cdots, n) \tag{2}$$

여기서 회귀함수 $g(x)$는 조건 $x_t = x$가 주어졌을 때 y_t의 조건부기대값을 나타내며, 오차항 $\{\epsilon_t\}$가 반드시 정규확률분포를 따르는 것은 아니다.

확률벡터 $[x_t, y_t]^t$의 결합확률밀도함수를 $f(x, y)$라고 하면, 다음 식이 성립한다.

$$g(x) = \int_{-\infty}^{\infty} y f(y|x) dy = \frac{\int_{-\infty}^{\infty} y f(x, y) dy}{f(x)} \tag{3}$$

식 (1.5.97)에서 알 수 있듯이, $f(x, y)$를 다음과 같은 커널평활확률밀도함수로 추정할 수 있다.

$$\hat{f}(x, y) \doteq \frac{1}{n} \sum_{t=1}^{n} K_{h_x}(x - x_t) K_{h_y}(y - y_t) \tag{4}$$

여기서 h_x와 h_y는 변수들 x와 y에 대응하는 대역폭들이다. 식 (4)를 식 (3)에 대입하면, 다음과 같은 회귀함수 $g(x)$의 커널평활추정량을 얻는다.

$$\hat{g}(x) \doteq \frac{\int_{-\infty}^{\infty} y \hat{f}(x, y) dy}{\hat{f}(x)} = \sum_{t=1}^{n} w_t y_t \tag{5}$$

여기서 w_t는 다음과 같다.

$$w_t \doteq \frac{K_{h_x}(x - x_t)}{\sum_{j=1}^{n} K_{h_x}(x - x_t)} \tag{6}$$

■

예제 1.5.53 Ait-Sahalia & Lo [6]는 평활확률밀도함수를 사용해서 위험중립확률밀도함수를 추정하였다. 이 예제에서는 이 추정에 대해 살펴보자.

Black-Scholes환경 하에서 유럽형콜옵션과 유럽형풋옵션의 무재정가치들 C_t^{BS}와 P_t^{BS}는

각각 다음 Black-Scholes식을 만족한다.

$$C_t^{BS} = S_t N(d_1) - Ke^{-r\tau} N(d_2) \tag{1}$$

$$P_t^{BS} = Ke^{-r\tau} N(-d_2) - S_t N(-d_1) \tag{2}$$

여기서 $\tau = T - t$이고, $N(x)$는 표준정규확률분포함수이며 d_1과 d_2는 각각 다음과 같다.

$$d_1 \doteq \frac{1}{\sigma\sqrt{\tau}} \left[\ln \frac{S_t}{Ke^{-r\tau}} + \frac{\sigma^2}{2}\tau \right] \tag{3}$$

$$d_2 \doteq \frac{1}{\sigma\sqrt{\tau}} \left[\ln \frac{S_t}{Ke^{-r\tau}} - \frac{\sigma^2}{2}\tau \right] = d_1 - \sigma\sqrt{\tau} \tag{4}$$

시장에서 제 j번째 유럽형콜옵션가격을 C_j라 하고, 이에 해당하는 원자산을 S_j, 만기까지 잔여기간을 τ_j 그리고 권리행사가격을 K_j라고 하자. 또한, 시장에는 유럽형콜옵션가격들 $C_1, C_2, \cdots, C_n$이 존재한다고 하자. 만약 원자산의 수익률이 기하Brown운동을 따르지 않으면, 유럽형콜옵션의 공정한 가치는 식 (1)과 다를 것이다. 이 가치를 $g(S_j, K_j, \tau_j)$라 하면, 다음 비선형회귀식이 성립한다고 할 수 있다.

$$C_j = g(S_j, K_j, \tau_j) + \epsilon_j, \quad (t = 1, 2, \cdots, n) \tag{5}$$

Breeden & Litzenberger [12]에서 알 수 있듯이, 만기시점에서 원자산의 확률밀도함수를 유럽형콜옵션가격들 $C_1, C_2, \cdots, C_n$의 선형결합으로 나타낼 수 있다. 따라서, 예제 1.5.52와 같은 방법을 사용해서, 비선형함수 g를 다음과 같이 추정할 수 있다.

$$\hat{g}(S, K, \tau) \doteq \sum_{j=1}^{n} w_j C_j \tag{6}$$

여기서 가중값 w_j는 다음과 같다.

$$w_j \doteq \frac{K_{h_S}(S - S_j) K_{h_K}(K - K_j) K_{h_\tau}(\tau - \tau_j)}{\sum_{j=1}^{n} K_{h_S}(S - S_j) K_{h_K}(K - K_j) K_{h_\tau}(\tau - \tau_j)} \tag{7}$$

단, h_x, h_y와 h_z는 변수들 , x, y와 z에 대응하는 대역폭들이다.

식 (6)의 비모수적 추정량 $\hat{g}(S, K, \tau)$를 사용해서 만기시점에서 위험중립확률밀도함수 $f_\tau(\cdot|S)$를 비모수적으로 추정할 수 있다. 위험중립확률밀도함수 $f_\tau(\cdot|S)$는 다음 Dupire식을

만족한다.

$$f_\tau(\cdot|S) = e^{r\tau}\frac{\partial^2 g(S,K,\tau)}{\partial K^2} \tag{8}$$

식 (6)을 식 (8)에 대입하면, 다음과 같이 위험중립확률밀도함수 $f_\tau(\cdot|S)$를 구할 수 있다.

$$\hat{f}_\tau(\cdot|S) = e^{r\tau}\frac{\partial^2 \hat{g}(S,K,\tau)}{\partial K^2} \tag{9}$$

■

제1.6절 이산형 확률분포

이 절에서는 자주 사용되는 이산형 확률분포에서 난수를 발생시키는 방법에 대해서 살펴보자.

1.6.1 이산형 일양확률분포

지지대가 $\{m, m+1, \cdots, n\}$인 이산형 일양확률분포의 확률질량함수는 다음과 같다.

$$f(x) = \frac{1}{n-m+1}1_{\{m,m+1,\cdots,n\}}(x) \tag{1.6.1}$$

이 이산형 일양확률분포의 평균과 분산은 각각 다음과 같다.

$$E(x) = \frac{n+m}{2}, \quad Var(x) = \frac{[n-m+1]^2-1}{12} \tag{1.6.2}$$

지지대가 $(0,1)$인 연속형 일양확률변수 u에 $n-m+1$을 곱한 $[n-m+1]u$의 정수부분에 m을 더한 $\lfloor m+[n-m+1]u \rfloor$은 지지대가 $\{m, m+1.\cdots, n\}$인 이산형 일양확률분포를 따른다. 여기서 $\lfloor a \rfloor$는 a의 정수부분을 의미한다. 이 성질을 이용해서 이산형 일양난수를 생성한다.

MATLAB함수 randi를 사용해서 이산형 일양난수를 생성할 수 있다. 다음 예제를 살펴보자.

예제 1.6.1 MATLAB함수 randi를 사용해서 이산형 일양난수를 발생시키는 방법을 살펴보기 위해서, 다음 MATLAB프로그램 RANDIexample101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: RANDIexample101.m
%  How to use randi.m
```

```
%  Programmed by CBS
%--------------------------------------------------------------------
clear all, close all
diary RANDIexample101.txt
rand('twister',5489)
u1 = randi(6)
u2 = randi(6,2)
u3 = randi(6,2,3)
u4 = randi(6,[2,3])
u5 = randi(6,2,3,2)
u6 = randi(6,[2,3,2])
X = ones(2,4);
u7 = randi(6,size(X))
u8 = randi([11,16],2,3)
defaultStream = RandStream.getGlobalStream;
savedState = defaultStream.State;
u9 = randi(10,1,5)
defaultStream.State = savedState;
u10 = randi(10,1,5)              % U10 =  u9
diary off
% End of program
% --------------------------------------------------------------------
```

첫 번째 명령문 'u1 = randi(6)'는 지지대가 $\{1,2,3,4,5,6\}$인 이산형 일양난수 1개를 생성해서 u1에 저장한다. 두 번째 명령문 'u2 = randi(6,2)'는 지지대가 $\{1,2,3,4,5,6\}$인 이산형 일양난수들로 이루어진 2×2행렬을 생성해서 u2에 저장한다. 세 번째 명령문 'u3 = randi(6,2,3)'는 지지대가 $\{1,2,3,4,5,6\}$인 이산형 일양난수들로 이루어진 행렬을 생성해서 u3에 저장한다. 네 번째 명령문 'u4 = randi(6,[2 3])'는 세 번째 명령문 'u3 = randi(6,2,3)'와 같다. 다섯 번째 명령문 'u5 = randi(6,2,3,2)'는 지지대가 인 이산형 일양난수들로 이루어진 행렬을 생성해서 u5에 저장한다. 여섯 번째 명령문 'u6 = randi(6,[2 3 2])'는 다섯 번째 명령문 'u5 = randi([2 3 2])'와 같다. 일곱 번째 명령문 'u7 = randi(6,size(X))'는 행렬 X와 같은 차원의 행렬을 생성해서 u7에 저장한다. 여덟 번째 명령문 'u8 = randi([11,16],2,3')'은 지지대가 인 이산형 일양난수들로 이루어진 행렬을 생성해서 u8에 저장한다. 나머지 MATLAB명령문들은 씨앗을 저장했다가 복원시키기 위한 것이다. 이렇게 생성된 u9과 u10은 동일하다. ■

예제 1.6.2 MATLAB을 이용해서 이산형 일양확률변수의 확률질량함수, 누적확률분포함수, 역누적확률분포함수 그리고 이산형 일양난수를 살펴보기 위해서, 다음 MATLAB프로그램 DiscreteUniformRV101.m을 실행해 보자.

```
% --------------------------------------------------------------------
%  Filename: DiscreteUniformRV101.m
%  Discrete Uniform Random Variable
%  Programmed by CBS
%--------------------------------------------------------------------
```

```
clear all, close all
n = 6                                   % Parameter
% PDF
yp1 = pdf('unid',1,n)
yp2 = unidpdf(1,n)
xx = 1:n
yy = pdf('unid',xx,n);
subplot(2,2,1)
stem(xx,yy,'r-s','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold', ...
    'xlim',[0.5 n+.5],'ylim',[0 0.32])
title('\bf PMF')
% CDF
yc1 = cdf('unid',1,n)
yc2 = unidcdf(1,n)
xx = 1:n
yy = cdf('unid',xx,n);
subplot(2,2,2)
stairs(xx,yy,'g-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold', ...
    'xlim',[0.5 n+0.5 ],'ylim',[0 1])
title('\bf CDF')
% Inverse CDF
yi1 = icdf('unid',10,n)
yi2 = unidinv(10,n)
xx = 0:0.01:1;
yy = icdf('unid',xx,n);
subplot(2,2,3)
plot(xx,yy,'b-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[1 n])
title('\bfInverse CDF')
% Random Numbers
yr1 = random('unid',n,3)
yr1 = unidrnd(n,3)
rand('twister',5489)
yran = random('unid',n,1,100);
subplot(2,2,4)
h1 = histogram(yran)
set(gca,'fontsize',11,'fontweigh','bold')
h1.NumBins = 20;
h1.FaceColor = [1 1 1];
h1.EdgeColor = 'black';
set(gca,'fontsize',11,'fontweigh','bold')
title('\bf Histogram')
saveas(gcf,'DiscreteUniformRV101','epsc')
save('DiscreteUniformRV101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 지지대가 $\{1,2,3,4,5,6\}$ 인 이산형 일양확률변수의 확률질량함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 100개 이산형 일양난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.6.1에 그려져 있다.

이산형 일양확률분포에 관한 MATLAB함수들로는 unidpdf, pdf, unidcdf, cdf, unidinv, icdf, unidstat, mle, unidrnd, random, randtool 등이 있다. ■

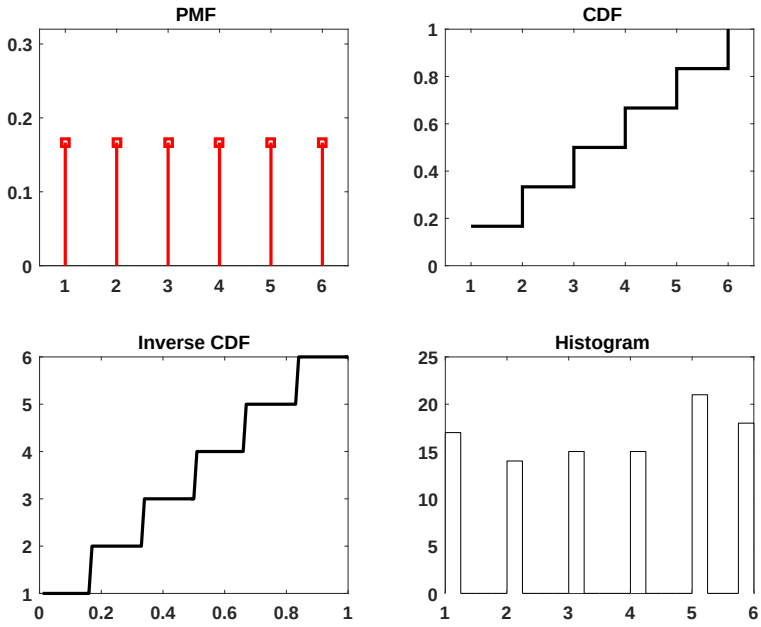


그림 1.6.1. 이산형 일양확률변수과 MATLAB

예제 1.6.3 R을 이용해서 이산형 일양확률변수의 확률질량함수, 누적확률분포함수, 역누적확률분포함수 그리고 이산형 일양난수를 살펴보기 위해서, 다음 R프로그램 DiscreteUniformRV101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------------
# Filename DiscreteUniformRV101R.R
# Discrete Uniform Random Variable
# Programmed by CBS
# ------------------------------------------------------------------------------
# Functions
dunifdisc <- function(x, minn, maxx)
             ifelse(x>=minn & x<=maxx & round(x)==x, 1/(maxx-minn+1),0)
punifdisc <- function(q, minn, maxx)
             ifelse(q<minn, 0, ifelse(q>=maxx, 1,(floor(q)-minn+1)/
                   (maxx-minn+1)))
qunifdisc <- function(p, minn, maxx) floor(p*(maxx-minn+1))
runifdisc <- function(n, minn, maxx) sample(minn:maxx, n, replace=T)
# Generating
Lower <- 1; Upper <- 6
nAbsissa <- 101
x <- seq(Lower+0.5,Upper+0.5,len=nAbsissa)
yp <- dunifdisc(x,Lower,Upper)         # PDF
yc <- punifdisc(x,Lower,Upper)         # CDF
ix <- seq(0.001,0.999,len=nAbsissa)
yi <- qunifdisc(ix,Lower,Upper)        # iCDF
nSim <- 100
set.seed(41)
yran <- runifdisc(nSim,Lower,Upper)       # random number

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
```

```
library(grid)
setEPS()
plot.new()
postscript('DiscreteUniformRV101R.eps')  # Start to save figure
RVdata1 <- data.frame(x,yp,yc)
RVdata2 <- data.frame(ix,yi)
RVdata3 <- data.frame(yran)
plot11 <- ggplot(RVdata1, aes(x=x,y=yp)) +
          geom_bar(stat="identity",fill="red",col="dark red") +
          xlab("x") +
          ggtitle("PDF")
plot12 <- ggplot(RVdata1, aes(x,yc)) +
          geom_line(col="green",lwd=1.2) +
          xlab("x") +
          ggtitle("CDF")
plot13 <- ggplot(RVdata2, aes(ix,yi)) +
          geom_line(col="blue",lwd=1.2) +
          xlab("x") +
          ggtitle("Inverse CDF")
plot14 <- ggplot(RVdata3, aes(x=yran)) +
          geom_histogram(bins=20,fill="white",color="black")+
          xlab("x") +
          ggtitle("Histogram")
pushViewport(viewport(layout=grid.layout(2,2)))
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot13, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot14, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()            # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 지지대가 $\{1,2,3,4,5,6\}$인 이산형 일양확률변수의 확률질량함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 100개 이산형 일양난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.6.2에 그려져 있다. ■

1.6.2 Bernoulli확률분포

지지대가 $\{0,1\}$인 Bernoulli확률분포의 확률질량함수는 다음과 같다.

$$f(x) = p^x[1-p]^{1-x}1_{\{0,1\}}(x) \tag{1.6.3}$$

이 Bernoulli확률분포의 평균과 분산은 각각 다음과 같다.

$$E(x) = p, \quad Var(x) = p[1-p] \tag{1.6.4}$$

지지대가 $(0,1)$인 연속형 일양확률변수 u에 대해서 확률변수 $1_{[0,p]}(u)$는 Bernoulli확률분

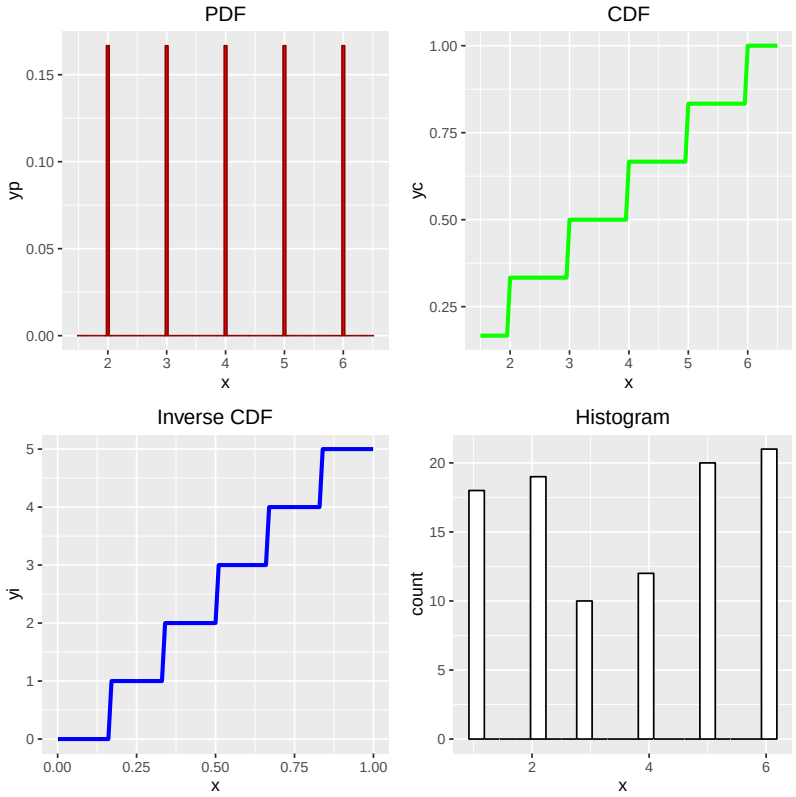


그림 1.6.2. 이산형 일양확률변수과 R

포를 따른다. 이 성질을 이용해서 Bernoulli난수를 생성한다.

예제 1.6.4 MATLAB을 이용해서 Bernoulli난수를 발생시키기 위해서, 다음 MATLAB프로그램 BernoulliRNgeneration101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: BernoulliRNgeneration101.m
%  BernoulliRandom Variable
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
diary BernoulliRNgeneration101.txt
p  = 1/3                                % Parameter
N = 100                                 % Number of RN's
yran = zeros(1,N);
rand('twister',5489)
yran = (rand(1,N) < p);
sumyran = sum(yran)
meanyran = mean(yran)
stdyran = std(yran)
diary off
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 모수가 $p = 1/3$인 Bernoulli확률분포에서 100개

난수들을 생성한다. 이중에서 관찰값이 1 인 개수는 31 이다. 따라서, 확률추정값은 $\hat{p} = 0.31$ 이고 또한 표본표준편차는 0.4648 이다. ■

예제 1.6.5 R을 이용해서 Bernoulli난수를 발생시키기 위해서, 다음 R프로그램 BernoulliRNgeneration101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------------
# Filename: BernoulliRNgeneration101R.R
# Bernoulli Random Number Generation
# Programmed by CBS
# ------------------------------------------------------------------------------
n <- 1; p <- 1/3
nSim <- 100
set.seed(13)
yran <- rbinom(nSim, size=n, prob=p)
( sumyran <- sum(yran) )
( meanyran <- mean(yran) )
( sdyran <- sd(yran) )
# ------------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 모수가 $p = 1/3$ 인 Bernoulli확률분포에서 100 개 난수들을 생성한다. 이중에서 관찰값이 1 인 개수는 28 이다. 따라서, 확률추정값은 $\hat{p} = 0.28$ 이고 또한 표본표준편차는 0.4513 이다. ■

1.6.3 이항확률분포

서로 독립이고 성공확률이 p 인 Bernoulli확률변수들 $v_1, v_2, \cdots, v_n$ 의 합 $x \doteq \sum_{i=1}^{n} v_i$ 를 이항확률변수라 부르고 $Binominal(n, p)$ 로 표기한다. 이항확률분포의 확률질량함수는 다음과 같다.

$$f(x) = \binom{n}{x} p^x [1-p]^{n-x} 1_{\{0,1,\cdots,n\}}(x) \tag{1.6.5}$$

이 이항확률분포의 평균과 분산은 각각 다음과 같다.

$$E(x) = np, \quad Var(x) = np[1-p] \tag{1.6.6}$$

Bernoulli난수를 이용해서 이항난수를 생성할 수 있다. 우선, 지지대가 $(0, 1)$ 인 일양난수들 n 개를 생성하고 이 중에서 p 보다 작은 일양난수들의 개수를 x 라 하면, x 는 이항확률분포를 따른다. 즉, 성공확률이 p 인 Bernoulli난수들 $v_1, v_2, \cdots, v_n$ 을 생성하면 그들의 합 $x = \sum_{i=1}^{n} v_i$ 은 이항난수이다.

역함수법을 적용하기 위해서 다음과 같이 이항확률분포함수를 정의하자.

$$F(x) = \sum_{k=0}^{\min\{\lfloor x \rfloor, n\}} \binom{n}{k} p^k [1-p]^{n-k} \tag{1.6.7}$$

이 이항확률분포의 역함수는 다음과 같다.

$$F^{-1}(x) = \inf\{v \mid F(v) \le x\} \tag{1.6.8}$$

식 (1.6.8)과 역함수법을 사용해서 이항난수들을 발생시킬 수 있다.

예제 1.6.6 식 (1.6.8)과 역함수법을 사용해서 이항난수들을 생성하기 위해서 다음 MATLAB 프로그램 InverseMethodBinomial101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: InverseMethodBinomial101.m
%  Weibull Random Numbers by Inverse Method
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
n = 20, p = 0.4                                % Parameters
N = 200                                        % Number of RN
rand('twister',5489)
yran = icdf('bino',rand(1,N),n,p)              % Inverse CDF
% Goodness-of-Fit Test
[H,P_value,stat] = chi2gof(yran,'cdf',{@binocdf,n,p})
% Plotting Histogram and PDF
xx = linspace(0,n,n+1);
Nheight = hist(yran,xx)/N;
bar(xx,Nheight,0.2,'w')
set(gca,'fontsize',11,'fontweigh','bold','xlim',[-0.5 n+0.5 ])
hold on
pp = binopdf(xx,n,p);                           % True PMF
plot(xx,pp,'r*','linewidth',2)
legend('Histogram','True PDF',1)
xlabel('\bf x'), ylabel('\bf Relative Frequency')
hold off
saveas(gcf,'InverseMethodBinomial101','epsc')
save('InverseMethodBinomial101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 모수벡터가 $(n, p) = (20, 0.4)$인 이항확률분포에서 난수들 200개를 발생한다. MATLAB함수 chi2gof.m를 사용해서 카이제곱검정을 한 결과, 검정통계량값은 2.3199이고 자유도는 6이며 p값이 0.8881으로 이 난수들이 이항확률분포에서 생성되었다는 귀무가설이 채택된다. 이 MATLAB프로그램을 실행하면, 그림 1.6.3 가 출력된다. 그림 1.6.3 에서 막대그래프는 이 난수들의 히스토그램이고, 적색 별표는 진짜확률

질량함수이다. 그림 1.6.3 에서 이 난수들이 이항확률분포로부터 발생되었다는 것을 확인할 수 있다. ■

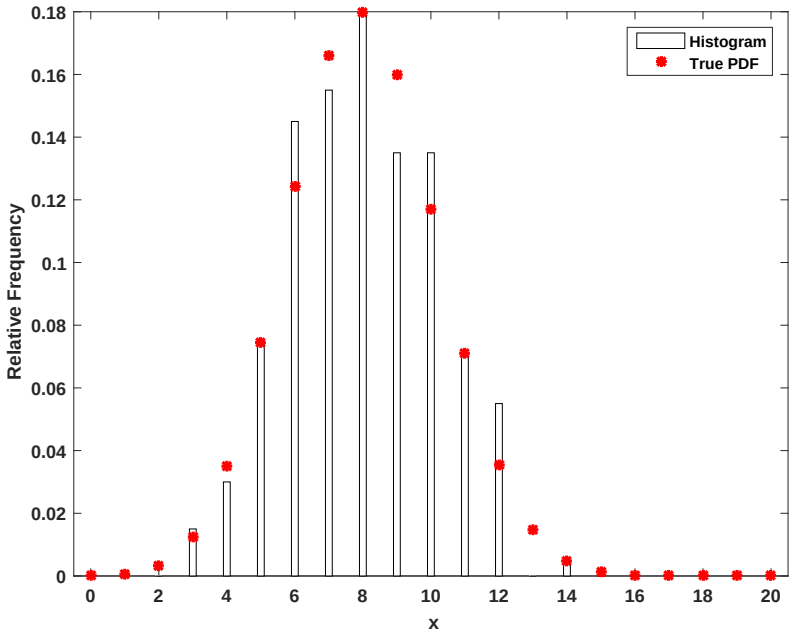


그림 1.6.3. 역함수법과 이항확률분포

예제 1.6.7 MATLAB을 이용해서 이항확률변수의 확률질량함수, 누적확률분포함수, 역누적확률분포함수 그리고 이항난수를 살펴보기 위해서, 다음 MATLAB프로그램 BinomialRV101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: BinomialRV101.m
%  Binomial Random Variable
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
n = 10, p = 0.6                              % Parameters
% PDF
yp1 = pdf('bino',1,n,p)
yp2 = binopdf(1,n,p)
xx = 1:n
yy = pdf('bino',xx,n,p);
subplot(2,2,1)
stem(xx,yy,'rs','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0.5 n+.5],'ylim',[0 0.32])
title('\bf PMF')
% CDF
yc1 = cdf('bino',1,n,p)
yc2 = binocdf(1,n,p)
xx = 1:n
yy = cdf('bino',xx,n,p);
subplot(2,2,2)
stairs(xx,yy,'g-','linewidth',2)
```

```
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0.5 n+0.5 ],'ylim',[0 1])
title('\bf CDF')
% Inverse CDF
yi1 = icdf('bino',10,n,p)
yi2 = binoinv(10,n,p)
xx = 0:0.01:1;
yy = icdf('bino',xx,n,p);
subplot(2,2,3)
plot(xx,yy,'b-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[1 n])
title('\bfInverse CDF')
% Random Numbers
yr1 = random('bino',n,p,3)
yr1 = binornd(n,p,3)
rand('twister',5489)
yran = random('bino',n,p,1,100);
subplot(2,2,4)
h1 = histogram(yran)
set(gca,'fontsize',11,'fontweigh','bold')
h1.NumBins = 20;
h1.FaceColor = [1 1 1];
h1.EdgeColor = 'black';
set(gca,'fontsize',11,'fontweigh','bold')
title('\bf Histogram')
saveas(gcf,'BinomialRV101','epsc')
save('BinomialRV101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 지지대가 $\{1, 2, \cdots, 10\}$ 인 이항확률변수의 확률질량함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 100 개 이항난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.6.4에 그려져 있다.

이항확률분포에 관한 MATLAB함수들로는 binopdf, pdf, binocdf, cdf, binoinv, icdf, binostat, binofit, mle, fitdist, dfittool, binornd, random, randtool 등이 있다. ■

예제 1.6.8 R을 이용해서 이항확률변수의 확률질량함수, 누적확률분포함수, 역누적확률분포함수 그리고 이항난수를 살펴보기 위해서, 다음 R프로그램 BinomialRV101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: BinomialRV101R.R
# Binomial Random Variable
# Programmed by CBS
# ------------------------------------------------------------------------
nAbsissa <- 101
n <- 10; p <- 0.6
x <- c(0:n)
yp <- dbinom(x, size=n, prob=p, log=FALSE)                    # PDF
yc <- pbinom(x, size=n, prob=p, lower.tail=TRUE, log.p=FALSE)  # CDF
ix <- seq(0,1,len=nAbsissa)
```

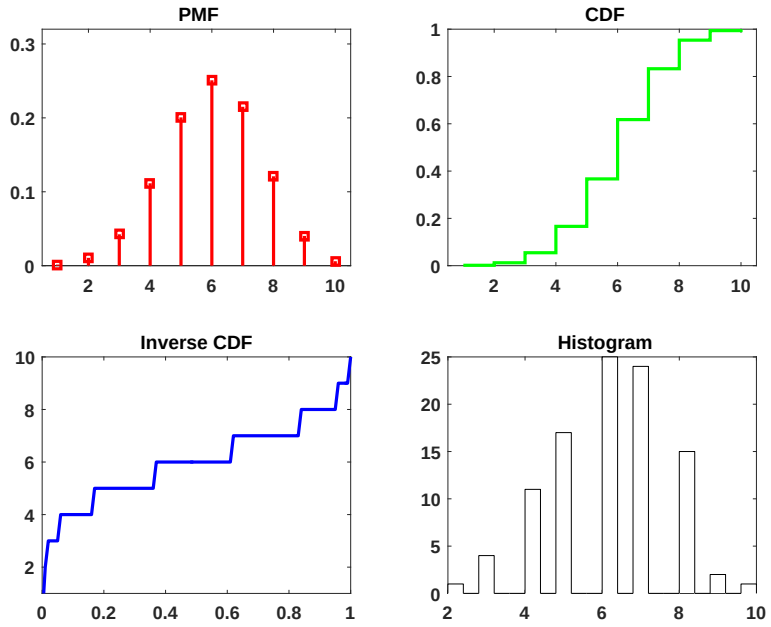


그림 1.6.4. 이항확률변수와 MATLAB

```
yi <- qbinom(ix, size=n, prob=p, lower.tail=TRUE, log.p=FALSE) #iCDF
nSim <- 100
set.seed(41)
yran <- rbinom(nSim, size=n, prob=p)

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('BinomialRV101R.eps')  # Start to save figure
RVdata1 <- data.frame(x,yp,yc)
RVdata2 <- data.frame(ix,yi)
RVdata3 <- data.frame(yran)
plot11 <- ggplot(RVdata1, aes(x,yp)) +
          geom_bar(stat="identity",fill="white",col="red") +
          geom_point(col="black",lwd=2) +
          xlab("x") +
          ggtitle("PDF")
plot12 <- ggplot(RVdata1, aes(x,yc)) +
          geom_step(col="green",lwd=1.2) +
          xlab("x") +
          ggtitle("CDF")
plot13 <- ggplot(RVdata2, aes(ix,yi)) +
          geom_step(col="blue",lwd=1.2) +
          xlab("x") + ylim(0,10) +
          ggtitle("Inverse CDF")
plot14 <- ggplot(RVdata3, aes(x=yran)) +
          geom_histogram(binwidth=0.25,fill="white",color="black")+
          xlab("x") +
          ggtitle("Histogram")
pushViewport(viewport(layout=grid.layout(2,2)))
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
```

```
print(plot13, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot14, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()             # End to save figure
# ------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 지지대가 $\{1, 2, \cdots, 10\}$인 이항확률변수의 확률질량함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 100개 이항난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 fig:BinomialRV101R에 그려져 있다. ■

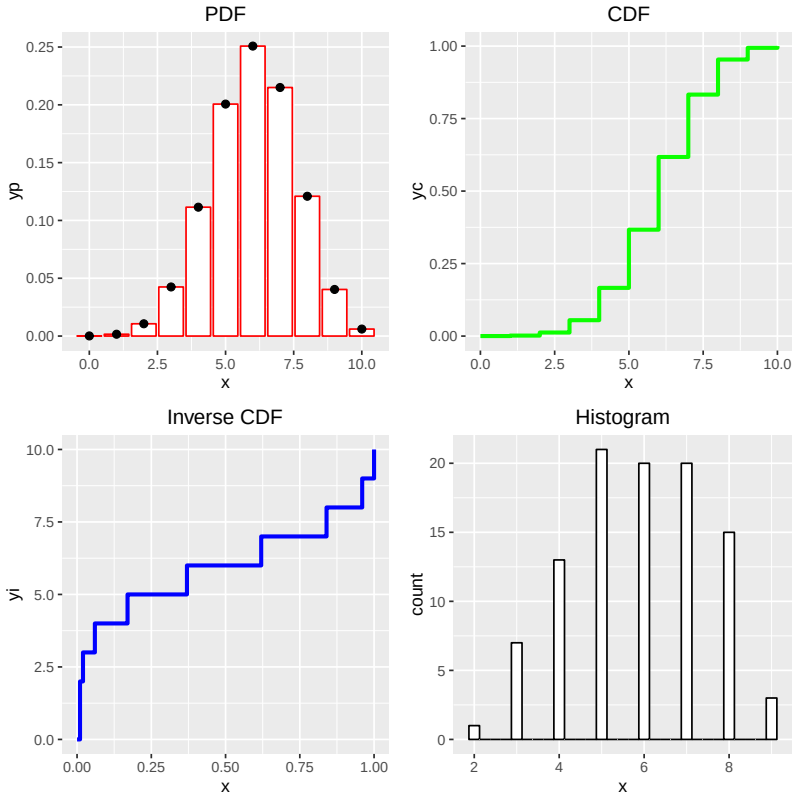


그림 1.6.5. 이항확률변수와 R

1.6.4 기하확률분포

성공확률이 p인 Bernouille시행에서 처음으로 성공할 때까지 필요한 실패횟수 x의 확률질량함수는 다음과 같다.

$$f(x) = p[1-p]^x 1_{\{0,1,\cdots\}}(x) \tag{1.6.9}$$

이 기하확률질량함수(geometric mass function)에 해당하는 기하확률분포함수는 다음과 같다.

$$F(x) = \left\{1 - [1-p]^{x+1}\right\} 1_{\{0,1,\cdots\}}(x) \tag{1.6.10}$$

이 기하확률분포의 평균과 분산은 각각 다음과 같다.

$$E(x) = \frac{1-p}{p}, \quad Var(x) = \frac{1-p}{p^2} \tag{1.6.11}$$

예제 1.6.9 역함수법을 사용해서 기하난수(geometric random number)를 생성해보자. 평균이 μ인 지수확률변수(exponential random variable)의 정수부분을 x로 표기하자. 지수확률분포에 따르는 확률변수 w는 다음 식을 만족한다.

$$Pr(x \le w) = \exp\left(-\frac{x}{\mu}\right) \tag{1}$$

다음 식들이 성립한다.

$$\begin{aligned} Pr(x = k) &= Pr(k \le w < k+1) = Pr(k \le w) - Pr(k+1 \le w) \\ &= e^{-k/u} - e^{-[k+1]/u} = e^{-k/u}[1 - e^{-1/u}] = [e^{-1/u}]^k[1 - e^{-1/u}] \end{aligned} \tag{2}$$

식 (1.6.9)와 식 (2)에서 알 수 있듯이, Bernouille시행의 성공확률을 $p = 1 - \exp(-1/\mu)$라 놓으면 확률변수 x는 기하확률질량함수 (1)을 갖는다. 따라서, 평균이 $\mu = -1/\ln(1-p)$인 지수확률변수의 정수부분은 기하확률질량함수를 갖는다. 지지대가 $(0,1)$인 일양확률변수 u에 대해서 다음과 같은 확률변수 x를 정의하자.

$$x \doteq \lfloor \ln u / \ln(1-p) \rfloor \tag{3}$$

이 x는 성공확률이 p인 기하확률변수이다.

지금까지 설명한 방법을 적용해서 기하난수들을 생성하기 위해서, 다음 MATLAB프로그램 InverseMethodGeometric101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: InverseMethodGeometric101.m
%  Generating Geometric Random Numbers
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
N = 1000;
```

```
p = 0.2, lnq = -0.223143551314210      % p = 0.2, lnq = ln(1-p)
rand('twister',5489)
yran = floor(-log(rand(1,N))/0.223143551314210)';
% Chi-Squared Test
xNum = 31;
xx = 0:1:xNum-1;
pHat = 1/(1+mean(yran))
expCounts = N*geopdf(xx,pHat)
[H,P_value,stat] = chi2gof(yran,'ctrs',xx,'expected',expCounts)
% Histogram
yNum = hist(yran,xx)
yFreq = yNum/N;
bar(xx,yFreq,0.5,'w')
set(gca,'fontsize',11,'fontweigh','bold')
hold on
ppxx = 0.2*0.8.^xx;                        % True PDF
plot(xx,ppxx,'r*','linewidth',2)
legend('\bf Histogram','\bf True PMF',1)
xlabel('x'), ylabel('Relative Frequency')
axis([-0.5 xNum-0.5 0 0.25 ])
hold off
saveas(gcf,'InverseMethodGeometric101','epsc')
save('InverseMethodGeometric101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 성공확률이 $p = 0.2$인 기하확률분포로부터 난수들 10000개가 생성된다. MATLAB함수 chi2gof을 사용해서 적합성검정을 한 결과, 카이제곱통계량은 16.3319이고 자유도는 17이며 p값이 0.5004로 이 난수들이 이 기하확률분포로부터 생성되었다는 귀무가설이 채택된다. MATLAB함수 chi2gof.m의 옵션 ctrs는 상자들(bins)의 중점들을 나타냄에 유의하라. 이 MATLAB프로그램을 실행하면, 그림 1.6.6가 출력된다. 그림 1.6.6에서 막대그래프는 이 난수들의 히스토그램이고, 적색 별표는 진짜확률질량함수이다. 그림 1.6.6에서 이 난수들이 기하확률분포로부터 발생되었다는 것을 확인할 수 있다. ■

예제 1.6.10 MATLAB을 이용해서 지수확률변수의 확률질량함수, 누적확률분포함수, 역누적확률분포함수 그리고 기하난수를 살펴보기 위해서, 다음 MATLAB프로그램 GeometricRV1.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: GeometricRV101.m
%  Geometric Random Variable
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
p = 0.2                                % Parameter
ndum = 20
% PDF
yp1 = pdf('geo',1,p)
yp2 = geopdf(1,p)
xx = 0:ndum
```

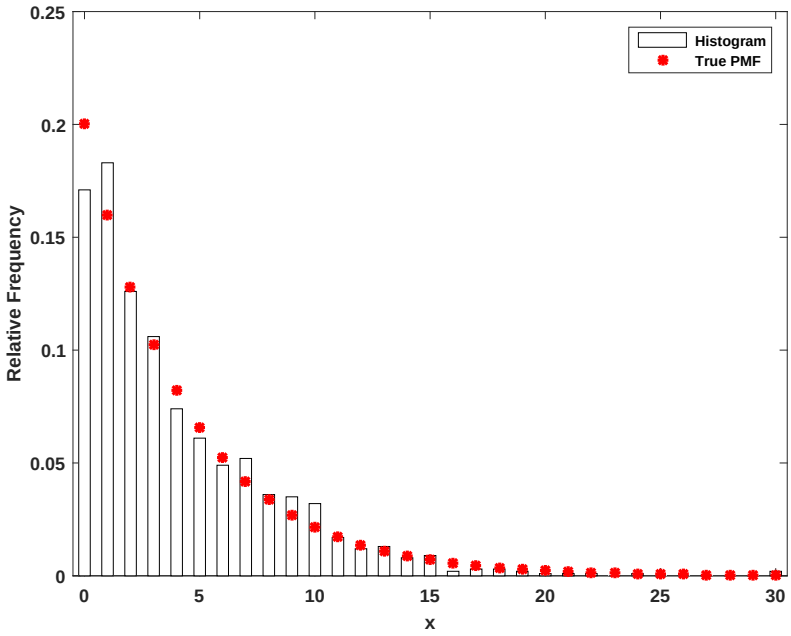


그림 1.6.6. 역함수법과 기하확률분포

```
yy = pdf('geo',xx,p);
subplot(2,2,1)
stem(xx,yy,'r--+','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0.5 ndum+.5])
title('\bf PMF')
% CDF
yc1 = cdf('geo',1,p)
yc2 = geocdf(1,p)
xx = 0:ndum
yy = cdf('geo',xx,p);
subplot(2,2,2)
stairs(xx,yy,'g-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold', ...
    'xlim',[0.5 ndum+0.5 ],'ylim',[0 1])
title('\bf CDF')
% Inverse CDF
yi1 = icdf('geo',10,p)
yi2 = geoinv(10,p)
xx = 0:0.01:1;
yy = icdf('geo',xx,p);
subplot(2,2,3)
plot(xx,yy,'b-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[1 ndum])
title('\bfInverse CDF')
% Random Numbers
yr1 = random('geo',p,3)
yr1 = geornd(p,3)
rand('twister',5489)
yran = random('geo',p,1,100);
subplot(2,2,4)
h1 = histogram(yran)
set(gca,'fontsize',11,'fontweigh','bold')
h1.NumBins = 20;
h1.FaceColor = [1 1 1];
h1.EdgeColor = 'black';
set(gca,'fontsize',11,'fontweigh','bold')
```

```
title('\bf Histogram')
saveas(gcf,'GeometricRV101','epsc')
save('GeometricRV101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 기하확률변수의 확률질량함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 100개 기하난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.6.7에 그려져 있다.

기하확률분포에 관한 MATLAB함수들로는 geopdf, pdf, geocdf, cdf, geoinv, icdf, geostat, mle, geornd, random, randtool 등이 있다. ■

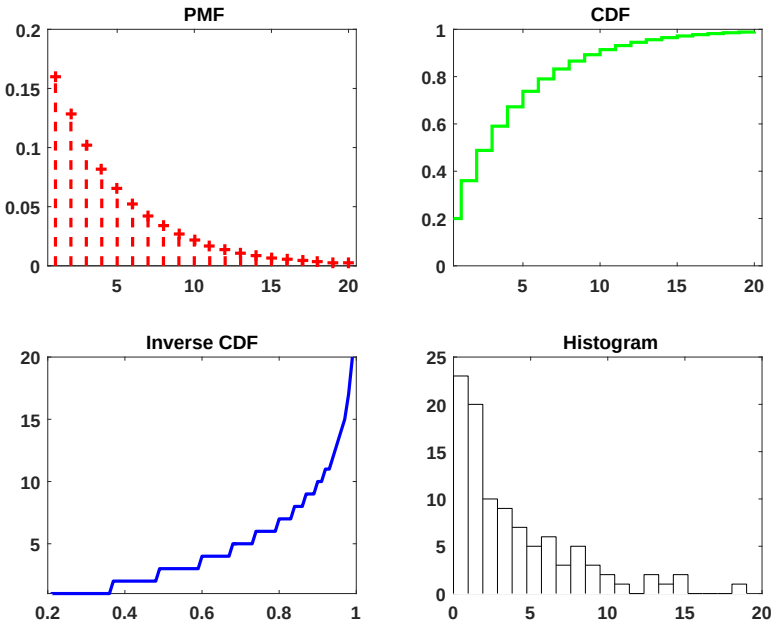


그림 1.6.7. 기하확률변수와 MATLAB

예제 1.6.11 R을 이용해서 지수확률변수의 확률질량함수, 누적확률분포함수, 역누적확률분포함수 그리고 기하난수를 살펴보기 위해서, 다음 R프로그램 GeometricRV1.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: GeometricRV101R.R
# Geometric Random Variable
# Programmed by CBS
# ------------------------------------------------------------------------
nAbsissa <- 101
p <- 0.2; ndum <- 20
x <- c(0:ndum)
yp <- dgeom(x, prob=p, log=FALSE)                      # PDF
yc <- pgeom(x, prob=p, lower.tail=TRUE, log.p=FALSE)   # CDF
ix <- seq(0,1,len=nAbsissa)
```

```
yi <- qgeom(ix, prob=p, lower.tail=TRUE, log.p=FALSE) #iCDF
nSim <- 100
set.seed(41)
yran <- rgeom(nSim, prob=p)

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('GeometricRV101R.eps')  # Start to save figure
RVdata1 <- data.frame(x,yp,yc)
RVdata2 <- data.frame(ix,yi)
RVdata3 <- data.frame(yran)
plot11 <- ggplot(RVdata1, aes(x,yp)) +
          geom_bar(stat="identity",fill="white",col="red") +
          geom_point(col="black",lwd=2) +
          xlab("x") +
          ggtitle("PDF")
plot12 <- ggplot(RVdata1, aes(x,yc)) +
          geom_step(col="green",lwd=1.2) +
          xlab("x") +
          ggtitle("CDF")
plot13 <- ggplot(RVdata2, aes(ix,yi)) +
          geom_step(col="blue",lwd=1.2) +
          xlab("x") + ylim(0,10) +
          ggtitle("Inverse CDF")
plot14 <- ggplot(RVdata3, aes(x=yran)) +
          geom_histogram(binwidth=0.25,fill="black",color="black")+
          xlab("x") +
          ggtitle("Histogram")
pushViewport(viewport(layout=grid.layout(2,2)))
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot13, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot14, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()            # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 기하확률변수의 확률질량함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 100개 기하난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.6.8에 그려져 있다.

기하확률분포에 관한 MATLAB함수들로는 geopdf, pdf, geocdf, cdf, geoinv, icdf, geostat, mle, geornd, random, randtool 등이 있다. ■

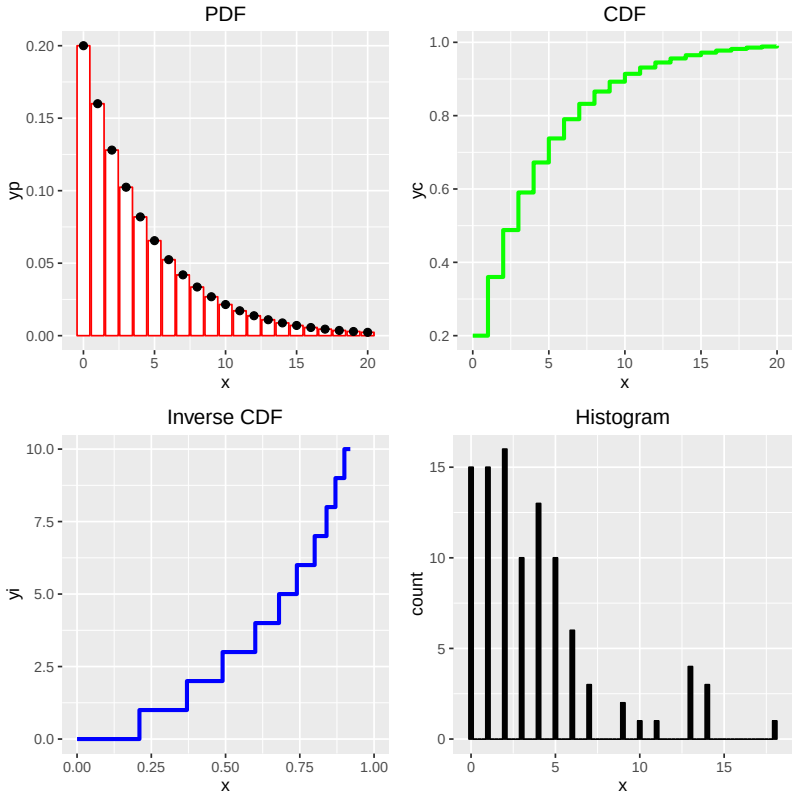


그림 1.6.8. 기하확률변수와 R

1.6.5 Pascal확률분포

성공확률이 p인 Bernouille시행에서 제 m번째 성공을 할 때까지 필요한 실험횟수 x의 확률질량함수는 다음과 같다.

$$f(x) = \binom{x-1}{m-1} p^m [1-p]^{x-m} 1_{\{m,m+1,\cdots\}}(x) \tag{1.6.12}$$

이 Pascal확률분포의 평균과 분산은 각각 다음과 같다.

$$E(x) = \frac{m}{p}, \quad Var(x) = \frac{m[1-p]}{p^2} \tag{1.6.13}$$

서로 독립이고 모수가 p인 기하확률변수들 $v_1, v_2, \cdots, v_m$에 대해서 확률변수 $x \doteq \sum_{k=1}^{m} [v_k + 1]$은 모수벡터가 $[p, m]$인 Pascal확률분포를 따른다. 이 성질을 이용해서 Pascal난수를 발생시킬 수 있다.

예제 1.6.12 지금까지 설명한 방법을 적용해서 Pascal난수들을 생성하기 위해서, 다음 MATLAB프로그램 PascalRandomNumber101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: PascalRandomNumber101.m
%  Generating Pascal Random Numbers
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
N = 1000;                                          % Number of RN's
m = 5, p = 0.5, lnq =  -0.693147180559945          % lnq = ln(1-p)
rand('twister',5489)
yran = sum(floor(-log(rand(m,N))/0.693147180559945)+1,1)';
% Chi-Squared Test
xNum = 31;
for jj=m:1:xNum-1
    expC(jj-m+1) = N*nchoosek(jj-1,m-1).*p^m.*(1-p).^(jj-m);
end
[H,P_value,stat] = chi2gof(yran,'edges',(m-.5):1:xNum-.5, ...
                           'expected',expC)
% Histogram
xx = m:1:xNum-1
yFreq = hist(yran,xx)/N;
bar(xx,yFreq,0.5,'w')
set(gca,'fontsize',11,'fontweigh','bold')
hold on
plot(xx,expC/N,'r*','linewidth',2)
legend('\bf Histogram','\bf True PMF',1)
xlabel('x'), ylabel('Relative Frequency')
axis([ m-0.5 xNum-0.5 0 0.16 ])
hold off
saveas(gcf,'PascalRandomNumber101','epsc')
save('PascalRandomNumber101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 성공확률이 $p = 0.5$인 Bernouille시행에서 $m = 5$번째 성공을 할 때까지 필요한 실험회수를 나타내는 Pascal난수들 1000개가 생성된다. MATLAB 함수 chi2gof을 사용해서 적합성검정을 한 결과, 카이제곱통계량은 5.3667이고 자유도는 15이며 p값이 0.9885이다. 따라서 이 난수들이 이 Pascal확률분포로부터 생성되었다는 귀무가설이 채택된다. 이 MATLAB프로그램을 실행하면, 그림 1.6.9이 출력된다. 그림 1.6.9에서 막대그래프는 이 난수들의 히스토그램이고, 적색 별표는 진짜확률질량함수이다. 그림 1.6.9에서 이 난수들이 Pascal분포로부터 발생되었다는 것을 확인할 수 있다. ■

1.6.6 음이항확률분포

성공확률이 p인 Bernouille시행에서 제 r번째 성공을 할 때까지 필요한 실패횟수 x의 확률질량함수는 다음과 같다.

$$f(x) = \binom{r+x-1}{r-1} p^r [1-p]^x \, 1_{\{0,1,\cdots\}}(x) \tag{1.6.14}$$

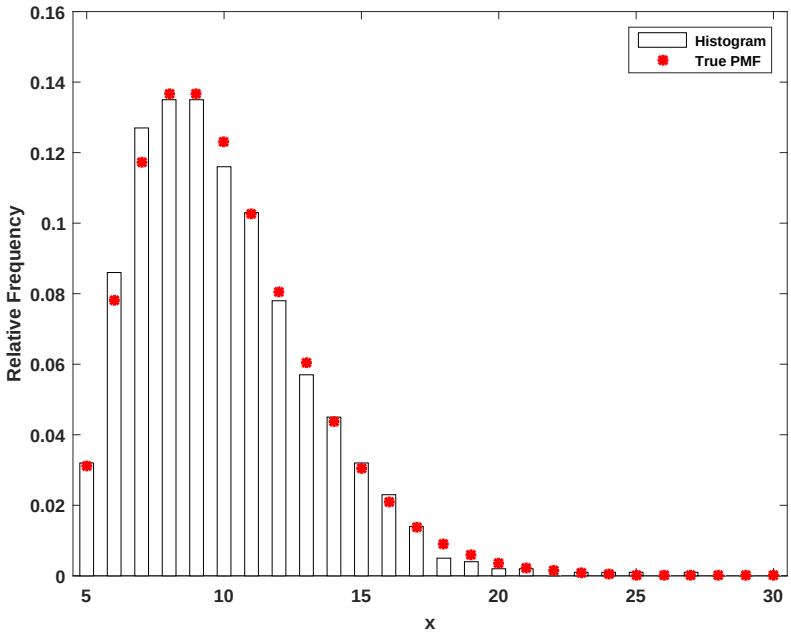


그림 1.6.9. Pascal확률분포

다음 식이 성립함을 쉽게 증명할 수 있다.

$$\binom{r+x-1}{r-1} = [-1]^x \binom{-r}{x} \tag{1.6.15}$$

따라서, 식 (1.6.14)에 해당하는 확률분포를 음이항확률분포(negative binomial distribution)이라 부른다. 이 음이항확률분포의 평균과 분산은 각각 다음과 같다.

$$E(x) = \frac{r[1-p]}{p}, \quad Var(x) = \frac{r[1-p]}{p^2} \tag{1.6.16}$$

서로 독립이고 모수가 p인 기하확률변수들 $v_1, v_2, \cdots, v_r$에 대해서 확률변수 $x \doteq \sum_{k=1}^{r} v_k$은 모수벡터가 $[p, r]$인 음이항확률분포를 따른다. 이 성질을 이용해서 음이항난수를 발생시킬 수 있다.

예제 1.6.13 MATLAB을 이용해서 음이항확률변수의 확률질량함수, 누적확률분포함수, 역누적확률분포함수 그리고 음이항난수를 살펴보기 위해서, 다음 MATLAB프로그램 NegativeBinomialRV101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: NegativeBinomialRV101.m
%  Negative Binomial Random Variable
%  Programmed by CBS
%------------------------------------------------------------------
```

```
clear all, close all
r = 10, p = 0.6                                % Parameters
% PDF
yp1 = pdf('nbin',1,r,p)
yp2 = nbinpdf(1,r,p)
xx = 0:20;
yy = pdf('nbin',xx,r,p);
subplot(2,2,1)
stem(xx,yy,'r','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 18])
title('\bf PMF')
% CDF
yc1 = cdf('nbin',1,r,p)
yc2 = nbincdf(1,r,p)
yy = cdf('nbin',xx,r,p);
subplot(2,2,2)
stairs(xx,yy,'g-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 18])
title('\bf CDF')
% Inverse CDF
yi1 = icdf('nbin',10,r,p)
yi2 = nbininv(10,r,p)
xx = 0:0.01:1;
yy = icdf('nbin',xx,r,p);
subplot(2,2,3)
plot(xx,yy,'b-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[0 18])
title('\bfInverse CDF')
% Random Numbers
yr1 = random('nbin',r,p,3)
yr1 = nbinrnd(r,p,3)
rand('twister',5489)
yran = random('nbin',r,p,1,100);
subplot(2,2,4)
h1 = histogram(yran)
set(gca,'fontsize',11,'fontweigh','bold')
h1.NumBins = 20;
h1.FaceColor = [1 1 1];
h1.EdgeColor = 'black';
set(gca,'fontsize',11,'fontweigh','bold')
title('\bf Histogram')
saveas(gcf,'NegativeBinomialRV101','epsc')
save('NegativeBinomialRV101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 음이항확률변수의 확률질량함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 100개 기하난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.6.10에 그려져 있다.

음이항확률분포에 관한 MATLAB함수들로는 nbinpdf, pdf, nbincdf, cdf, nbininv, icdf, nbinstat, nbinfit, mle, fitdist, dfittool, nbinrnd, random, randtool 등이 있다. ■

예제 1.6.14 R을 이용해서 음이항확률변수의 확률질량함수, 누적확률분포함수, 역누적확률분포함수 그리고 음이항난수를 살펴보기 위해서, 다음 R프로그램 NegativeBinomi-

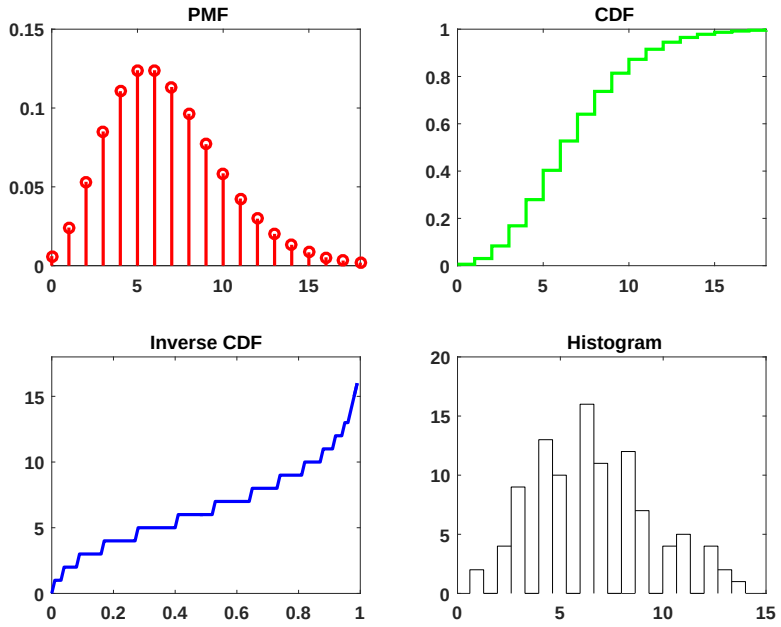


그림 1.6.10. 음이항확률변수와 MATLAB

alRV101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------------
# Filename: NegativeBinomialRV101R.R
# Negative Binomial Random Variable
# Programmed by CBS
# ------------------------------------------------------------------------------
nAbsissa <- 101
r <- 10; p <- 0.6;
x <- c(0:18)
yp <- dnbinom(x, size=r, prob=p, log = FALSE)                    # PDF
yc <- pnbinom(x, size=r, prob=p, lower.tail=TRUE, log.p=FALSE)  # CDF
ix <- seq(0,1,len=nAbsissa)
yi <- qnbinom(ix, size=r, prob=p, lower.tail=TRUE, log.p=FALSE) #iCDF
nSim <- 100
set.seed(41)
yran <- rnbinom(nSim, size=r, prob=p)

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('NegativeBinomialRV101R.eps')  # Start to save figure
RVdata1 <- data.frame(x,yp,yc)
RVdata2 <- data.frame(ix,yi)
RVdata3 <- data.frame(yran)
plot11 <- ggplot(RVdata1, aes(x,yp)) +
          geom_bar(stat="identity",fill="white",col="red") +
          geom_point(col="black",lwd=2) +
          xlab("x") +
          ggtitle("PDF")
plot12 <- ggplot(RVdata1, aes(x,yc)) +
```

```
          geom_step(col="green",lwd=1.2) +
          xlab("x") +
          ggtitle("CDF")
plot13 <- ggplot(RVdata2, aes(ix,yi)) +
          geom_step(col="blue",lwd=1.2) +
          xlab("x") + ylim(0,10) +
          ggtitle("Inverse CDF")
plot14 <- ggplot(RVdata3, aes(x=yran)) +
          geom_histogram(binwidth=0.25,fill="black",color="black")+
          xlab("x") +
          ggtitle("Histogram")
pushViewport(viewport(layout=grid.layout(2,2)))
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot13, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot14, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()             # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 음이항확률변수의 확률질량함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 100 개 난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.6.11에 그려져 있다. ■

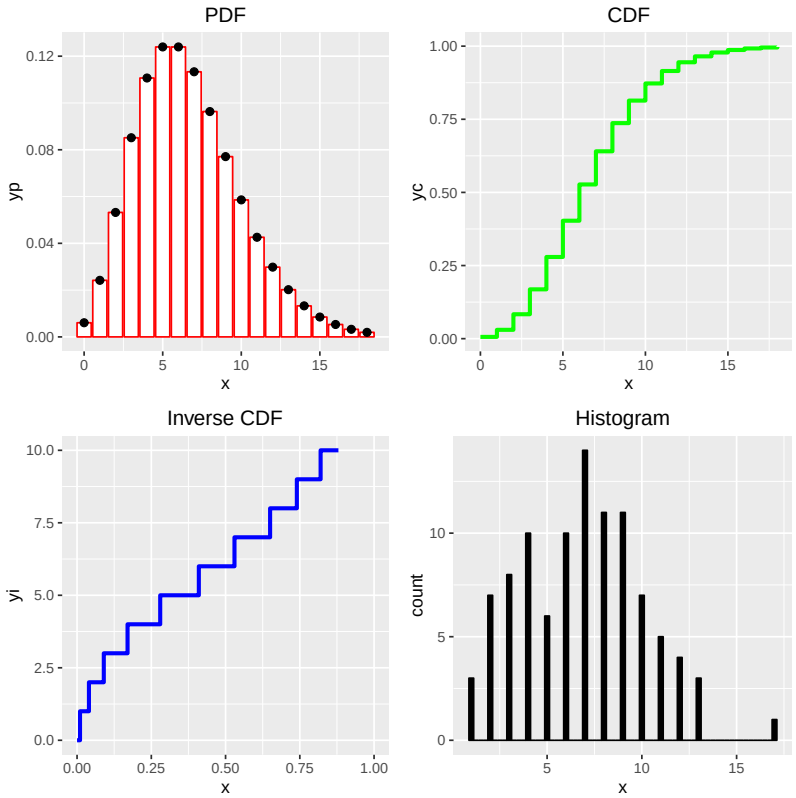


그림 1.6.11. 음이항확률변수와 R

1.6.7 Poisson확률분포

서로 독립적인 사건들이 발생하는 시간간격이 지수확률분포를 따를 때, 어떤 시간구간에서 이 사건이 발생하는 횟수는 Poisson확률분포를 따른다. 강도모수(intensity)가 $\lambda(>0)$인 Poisson확률분포의 확률질량함수는 다음과 같다.

$$f(x) = e^{-\lambda}\frac{\lambda^x}{x!}1_{\{0,1,\cdots\}}(x) \tag{1.6.17}$$

이 Poisson확률분포의 평균과 분산은 각각 다음과 같다.

$$E(x) = \lambda, \quad Var(x) = \lambda \tag{1.6.18}$$

예제 1.6.15 모수가 λ인 Poisson확률분포로부터 난수들을 생성하기로 하자. 서로 독립인 발생간격이 평균이 μ인 지수확률분포를 따르면, 시각 t까지 사건이 발생하는 횟수 x는 모수가 t/μ인 Poisson확률분포에 따른다. 즉, 다음 식이 성립한다.

$$Pr(x=k) = \frac{1}{k!}\frac{t^k}{\mu^k}\exp\left(-\frac{t}{\mu}\right), \quad (k=0,1,2,\cdots) \tag{1}$$

만약 $\mu=1$이고 $t=\lambda$이면, 확률변수 x는 모수가 λ인 Poisson확률분포를 따른다. 평균이 1인 지수확률분포에서 발생된 난수들 $\{e_j \mid j=1,2,\cdots\}$에 대해서, 확률변수 x를 다음과 같이 정의하자.

$$\sum_{j=1}^{k} e_j \le \lambda < \sum_{j=1}^{k+1} e_j \quad \Rightarrow \quad x=k \tag{2}$$

일양난수 u_j에 대해 $e_j = -\ln u_j$는 지수난수이다. 따라서, 식 (2)를 다음과 같이 쓸 수 있다.

$$\prod_{j=1}^{k} u_j \ge e^{-\lambda} > \prod_{j=1}^{k+1} u_j \quad \Rightarrow \quad x=k \tag{3}$$

식 (3)에서 알 수 있듯이, 일양난수를 차례로 곱해서 처음으로 $e^{-\lambda}$보다 작아지기 직전의 개수를 x라 하면, x는 모수 λ의 Poisson확률분포를 따른다.

지금까지 설명한 방법을 적용해서 Poisson난수들을 생성하기 위해서, 다음 MATLAB 프로그램 InverseMethodPoisson101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: InverseMethodPoisson101.m
%  Generating Poisson Random Numbers
```

```
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
N = 10000
lambda = 2;
elam = exp(-lambda)       % exp(-lambda) = 0.135335283236613
rand('twister',5489)
for jj = 1:N
    MrandNew= rand(1,1);
    kk = 0;
    while 12345
          if MrandNew < 0.135335283236613
             yran(jj) = kk; break
          else
             MrandOld = MrandNew;
             MrandNew = MrandOld*rand(1,1);
             kk = kk+1;
          end
    end
end
% Chi-Squared Test
xNum = 16;
xx = 0:1:xNum-1;
expCounts = N*poisspdf(xx,lambda)
[H,P_value,stat] = chi2gof(yran,'ctrs',xx,'expected',expCounts)
% Histogram
yFreq = hist(yran,xx)/N;
bar(xx,yFreq,0.3,'w')
set(gca,'fontsize',11,'fontweigh','bold')
pp = exp(-2).*2.^xx./factorial(xx);
hold on
plot(xx,pp,'rd','linewidth',2)
legend('\bf Histogram','\bf True PMF',1)
xlabel('x'), ylabel('Relative Frequency')
axis([ -0.5 xNum-0.5 0 0.35 ])
hold off
saveas(gcf,'InverseMethodPoisson101','epsc')
save('InverseMethodPoisson101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 모수가 $\lambda = 2$인 Poisson확률분포로부터 난수들 10000개가 생성된다. MATLAB함수 chi2gof.m을 사용해서 적합성검정을 한 결과, 카이제곱통계량은 4.4941이고 자유도는 8이며 p값이 0.8100이다. 따라서 이 난수들이 이 Poisson확률분포로부터 생성되었다는 귀무가설이 채택된다. 이 MATLAB프로그램을 실행하면, 그림 1.6.8이 출력된다. 그림 1.6.8에서 막대그래프는 이 난수들의 히스토그램이고, 적색 별표는 진짜확률질량함수이다. 그림 1.6.8에서 이 난수들이 Poisson확률분포로부터 발생되었다는 것을 확인할 수 있다. ■

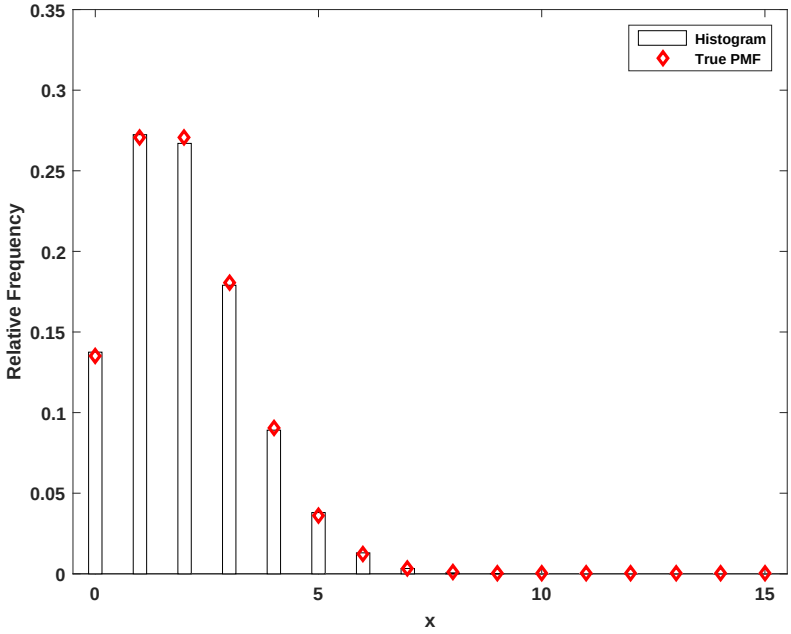


그림 1.6.12. 역함수법과 Poisson확률분포

예제 1.6.16 다음과 같은 Poisson확률질량함수에서 난수를 생성하기로 하자.

$$p_i = e^{-2}\frac{1}{i!}2^i, \quad (i = 0, 1, 2, \cdots) \tag{1}$$

보조확률질량함수로서 다음과 같은 기하확률질량함수를 사용하기로 하자.

$$q_i = a[1-a]^i, \quad (i = 0, 1, 2, \cdots) \tag{2}$$

충분히 큰 i에 대해서 다음 식들이 성립한다.

$$\frac{p_{i+1}}{p_i} \approx \frac{3}{i+1}, \quad \frac{q_{i+1}}{q_i} = 1-a \tag{3}$$

따라서 $i = 3$을 선택하면, $a = 1/4$가 좋을 것 같다. 다음 식이 성립한다.

$$\max_{0\le i\le 10}\frac{p_i}{q_i} = 1.9248 \tag{4}$$

즉, 상수 c로 1.93을 사용하기로 하자. 만약 일양난수 u가 $p_y/[cq_y]$보다 작으면, 이 y를 이산확률질량함수 $\{p_i\}$에서 생성된 난수 x로 채택한다. 그렇지 않으면, 다시 이산확률질량함수 $\{q_i\}$에서 난수 y를 생성한다.

지금까지 설명한 방법을 적용해서 이산형 확률분포로부터 난수들을 생성하기 위해서, 다음

MATLAB프로그램 AcceptRejectPoisson101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: AcceptRejectPoisson101.m
%  Generating Poisson Random Numbers
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
N = 10000;
pp = @(x) exp(-2)*2.^x./factorial(x);
qq = @(x) 0.25*0.75.^x;                          % a = 0.25
% Decide the constant c
c = ceil(max(pp(0:1:10)./qq(0:1:10))*100)/100
rand('twister',5489)
for jj = 1:N
    kk = 0;
    while 67890
       kk = kk + 1;
       ii = floor(log(rand(1,1))/log(0.75)) ;    % 1-a = 0.75
       if rand(1,1) < pp(ii)/(c*qq(ii))
          yran(jj) = ii; cc(jj) = kk; break
       end
    end
end
cmean = mean(c)       % averge number of steps until success
AcceptProb = 1/cmean  % acceptance probability
mean(yran), std(yran)
% Plotting Histogram and True PDF
binNum = 9
bin = 0:1:binNum-1;
yNum = hist(yran,bin)
yFreq = yNum/N;
bar(bin,yFreq,0.1,'w')
set(gca,'fontsize',11,'fontweigh','bold')
hold on
ppbin = pp(bin);
plot(bin,ppbin,'r*','linewidth',2)
legend('\bf Histogram','\bf True PDF',1)
xlabel('x'), ylabel('Relative Frequency')
axis([-0.5 binNum-0.5 0 0.35 ])
hold off
% Chi-squared Goodness-of-Fit test of Poisson PDF
lambdaHat = sum(bin.*yNum)/N;
expCounts = N*poisspdf(bin,lambdaHat)
[H,pvalue,stat] = chi2gof(bin,'ctrs',bin,'frequency',yNum, ...
                          'expected',expCounts,'nparams',1)
saveas(gcf,'AcceptRejectPoisson101','epsc')
save('AcceptRejectPoisson101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 확률질량함수 $\{p_i\}$에서 10000개 난수들을 생성한다. 채택기각법을 사용해서 확률질량함수 $\{p_i\}$로부터 난수 1개를 생성하기 위해서, 일양확률질량함수 $\{q_i\}$에서 평균 $\hat{c} = 1.93$개의 난수들을 생성하였다. 즉, 채택율은 0.5181이다. 이렇게 발생시킨 난수들의 히스토그램이 그림 1.6.13에 그려져 있다. 그림 1.6.13에서 막대그래프

는 이 난수들의 히스토그램이고, 적색 별표는 진짜확률질량함수이다. 그림 1.6.13에서 이 난수들의 히스토그램은 Poisson확률질량함수로부터 많이 떨어져 있지 않다. 이 난수들이 Poisson확률질량함수 $\{p_i\}$에서 발생되었는지를 검정하기 위해서 카이제곱검정을 한 결과, 카이제곱통계량값은 12.4871이고 자유도는 7이며 p값은 0.0856이다. 즉, 이 난수들이 Poisson 확률질량함수 $\{p_i\}$에서 발생되었다는 귀무가설을 채택한다. ■

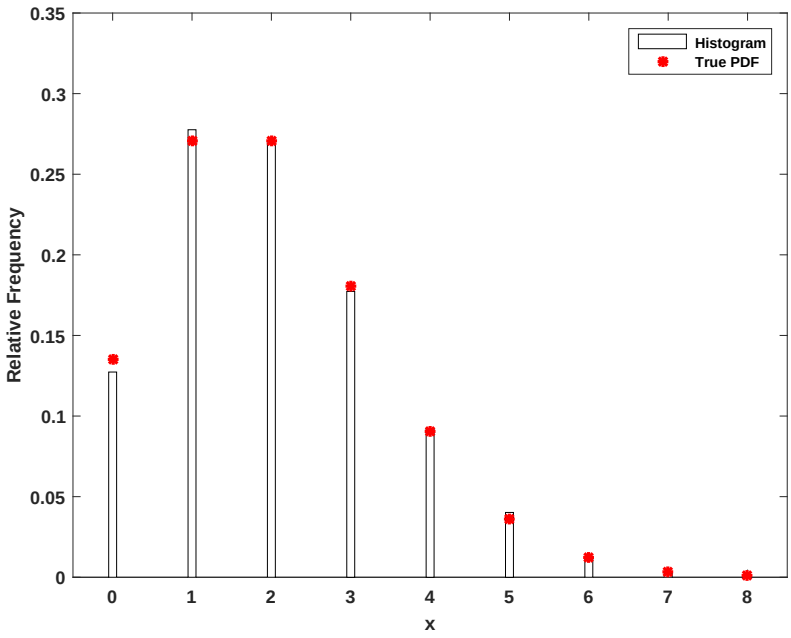


그림 1.6.13. 채택기각법과 Poisson확률분포

예제 1.6.17 MATLAB을 이용해서 Poisson확률변수의 확률질량함수, 누적확률분포함수, 역누적확률분포함수 그리고 음이항난수를 살펴보기 위해서, 다음 MATLAB프로그램 PoissonRV101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: PoissonRV101.m
%  Poisson Random Variable
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
lambda = 3                                % Parameter
% PDF
yp1 = pdf('poiss',1,lambda)
yp2 = poisspdf(1,lambda)
xx = 0:1:10;
yy = pdf('Poisson',xx,lambda);
subplot(2,2,1)
stem(xx,yy,'r','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[0 0.3])
title('\bf PMF')
% CDF
```

```
yc1 = cdf('poiss',1,lambda)
yc2 = poisscdf(1,lambda)
yy = cdf('Poisson',xx,lambda);
subplot(2,2,2)
stairs(xx,yy,'g','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[0 1])
title('\bf CDF')
% Inverse CDF
yi1 = icdf('poiss',0.5,lambda)
yi2 = poissinv(0.5,lambda)
xx = 0:0.01:1;
yy = icdf('Poisson',xx,lambda);
subplot(2,2,3)
plot(xx,yy,'b-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
title('\bfInverse CDF')
% Random Numbers
yr1 = random('poiss',lambda,3)
yr1 = poissrnd(lambda,3)
rand('twister',5489)
yran = random('Poisson',lambda,1,100);
subplot(2,2,4)
h1 = histogram(yran)
set(gca,'fontsize',11,'fontweigh','bold')
h1.NumBins = 20;
h1.FaceColor = [1 1 1];
h1.EdgeColor = 'black';
title('\bf Histogram')
saveas(gcf,'PoissonRV101','epsc')
save('PoissonRV101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, Poisson확률변수의 확률질량함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 100개 Poisson난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.6.14에 그려져 있다.

Poisson확률분포에 관한 MATLAB함수들로는 poisspdf, pdf, poisscdf, cdf, poissinv, icdf, poissstat, poissfit, mle, fitdist, dfittool, poissrnd, random, randtool 등이 있다. ■

예제 1.6.18 R을 이용해서 Poisson확률변수의 확률질량함수, 누적확률분포함수, 역누적확률분포함수 그리고 음이항난수를 살펴보기 위해서, 다음 R프로그램 PoissonRV101R.m을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: PoissonRV101R.R
# Poisson Random Variable
# Programmed by CBS
# ------------------------------------------------------------------------
nAbsissa <- 101
lambdaa <- 3
x <- c(0:10)
yp <- dpois(x, lambda=lambdaa, log = FALSE)                    # PDF
```

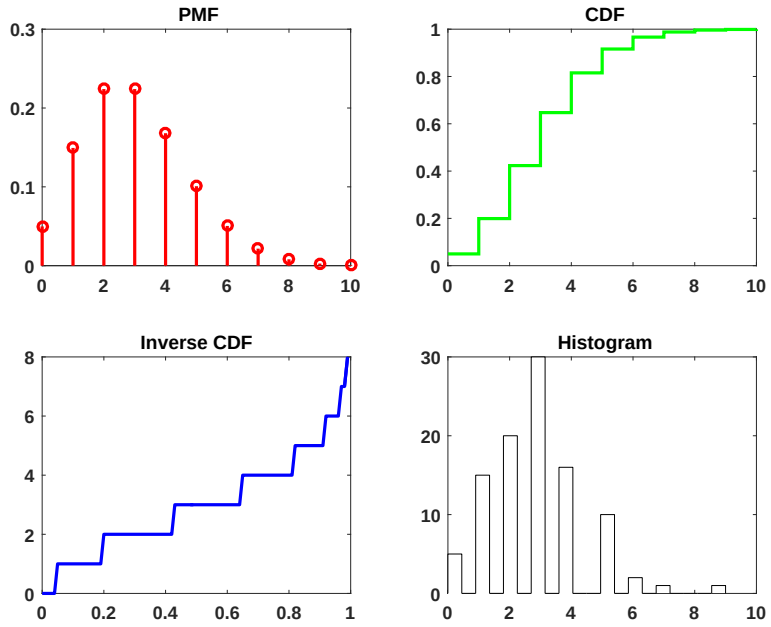


그림 1.6.14. Poisson확률변수와 MATLAB

```
yc <- ppois(x, lambda=lambdaa, lower.tail=TRUE, log.p=FALSE)  # CDF
ix <- seq(0,1,len=nAbsissa)
yi <- qpois(ix, lambda=lambdaa, lower.tail=TRUE, log.p=FALSE) #iCDF
nSim <- 100
set.seed(41)
yran <- rpois(nSim, lambda=lambdaa)

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('PoissonRV101R.eps')  # Start to save figure
RVdata1 <- data.frame(x,yp,yc)
RVdata2 <- data.frame(ix,yi)
RVdata3 <- data.frame(yran)
plot11 <- ggplot(RVdata1, aes(x,yp)) +
          geom_bar(stat="identity",fill="white",col="red") +
          geom_point(col="black",lwd=2) +
          xlab("x") +
          ggtitle("PDF")
plot12 <- ggplot(RVdata1, aes(x,yc)) +
          geom_step(col="green",lwd=1.2) +
          xlab("x") +
          ggtitle("CDF")
plot13 <- ggplot(RVdata2, aes(ix,yi)) +
          geom_step(col="blue",lwd=1.2) +
          xlab("x") + ylim(0,10) +
          ggtitle("Inverse CDF")
plot14 <- ggplot(RVdata3, aes(x=yran)) +
          geom_histogram(binwidth=0.25,fill="black",color="black")+
          xlab("x") +
          ggtitle("Histogram")
pushViewport(viewport(layout=grid.layout(2,2)))
```

```
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot13, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot14, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()             # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, Poisson확률변수의 확률질량함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 100 개 Poisson난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.6.15에 그려져 있다. ■

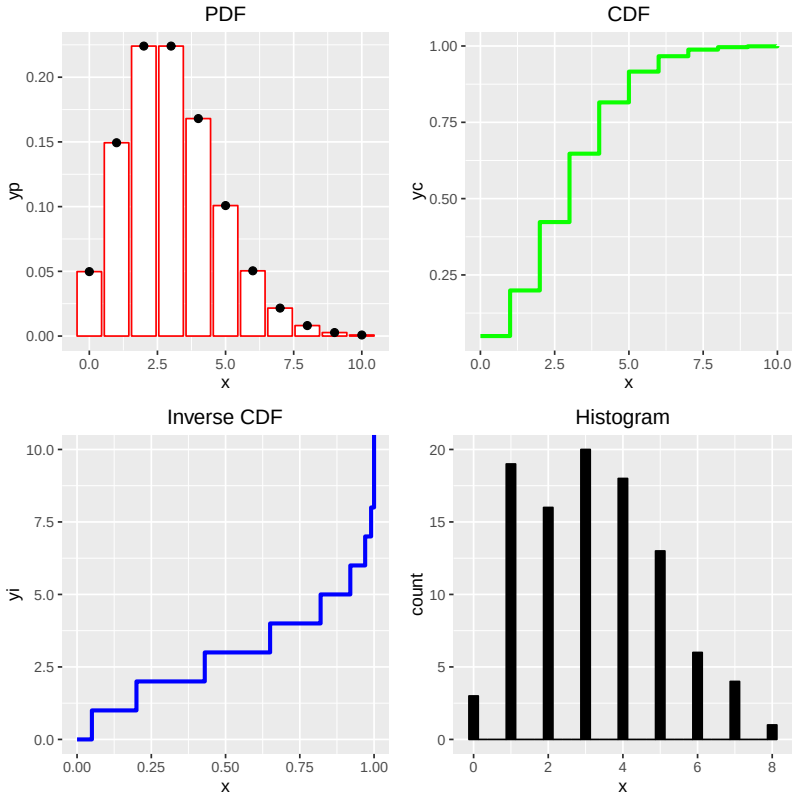


그림 1.6.15. Poisson확률변수와 R

1.6.8 초기하확률분포

전체 인구가 M 명이고 그 중 여자가 K 명이라고 하자. 전체에서 N 명을 뽑을 때 그 중 x 명이 여자일 확률은 다음과 같다.

$$f(x) = \frac{\binom{K}{x}\binom{M-K}{N-x}}{\binom{M}{N}} 1_{\{0,1,\cdots,N\}}(x) \tag{1.6.19}$$

식 (1.6.19) 의 확률밀도함수를 갖는 확률변수를 초기하확률변수 (hypergeometric random variable)라 부른다. 이 초기하확률분포의 평균과 분산은 각각 다음과 같다.

$$E(x) = N\frac{K}{M}, \quad Var(x) = N\frac{K[M-K][M-N]}{M^2[M-1]} \tag{1.6.20}$$

예제 1.6.19 MATLAB을 이용해서 초기하확률변수의 확률질량함수, 누적확률분포함수, 역누적확률분포함수 그리고 초기하난수를 살펴보기 위해서, 다음 MATLAB프로그램 HypergeometricRV101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: HypergeometricRV101.m
%  Hypergeometric Random Variable
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
M = 100, K = 20, N = 30                          % Parameters
x = 10
% PDF
yp1 = pdf('hyge',x,M,K,N)
yp2 = hygepdf(x,M,K,N)
xx = 0:1:x;
yy = pdf('Hypergeometric',xx,M,K,N);
subplot(2,2,1)
stem(xx,yy,'r','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
title('\bf PMF')
% CDF
yc1 = cdf('hyge',x,M,K,N)
yc2 = hygecdf(x,M,K,N)
yy = cdf('Hypergeometric',xx,M,K,N);
subplot(2,2,2)
stairs(xx,yy,'k','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
title('\bf CDF')
% Inverse CDF
yi1 = icdf('hyge',0.5,M,K,N)
yi2 = hygeinv(0.5,M,K,N)
xx = 0.01:0.01:0.99;
yy = icdf('Hypergeometric',xx,M,K,N);
subplot(2,2,3)
plot(xx,yy,'k-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
title('\bfInverse CDF')
% Random Numbers
rand('twister',5489)
yr1 = random('hyge',M,K,N,3)
yr1 = hygernd(M,K,N,3)
yran = random('Hypergeometric',M,K,N,1,100);
subplot(2,2,4)
h1 = histogram(yran)
set(gca,'fontsize',11,'fontweigh','bold')
h1.NumBins = 20;
h1.FaceColor = [1 1 1];
h1.EdgeColor = 'black';
```

```
title('\bf Histogram')
saveas(gcf,'HypergeometricRV101','epsc')
save('HypergeometricRV101','yran')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 초기하확률변수의 확률질량함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 100개 초기하난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.6.16에 그려져 있다.

초기하확률분포에 관한 MATLAB함수들로는 hygepdf, pdf, hygecdf, cdf, hygeinv, icdf, hygestat, hygernd, random, randtool 등이 있다. ■

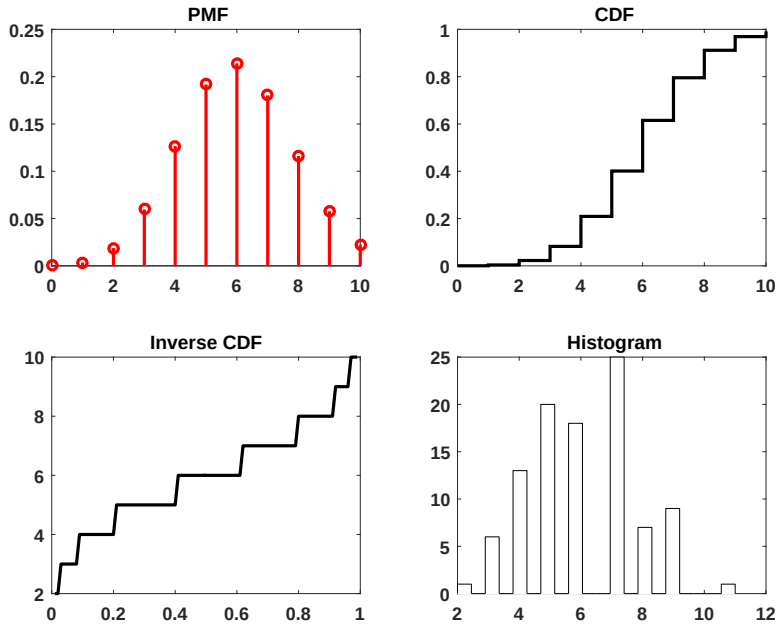


그림 1.6.16. 초기하확률변수와 MATLAB

예제 1.6.20 R을 이용해서 초기하확률변수의 확률질량함수, 누적확률분포함수, 역누적확률분포함수 그리고 초기하난수를 살펴보기 위해서, 다음 R프로그램 HypergeometricRV101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: HypergeometricRV101R.R
# Hypergeometric Random Variable
# Programmed by CBS
# ------------------------------------------------------------------------
nAbsissa <- 101
M <- 100; K <- 20; N <- 30; xmax <- 10
# m <- K; n <- M-K; k <- N
x <- c(0:xmax)
yp <- dhyper(x, m=K, n=M-K, k=N, log = FALSE)                    # PDF
```

```
yc <- phyper(x, m=K, n=M-K, k=N, lower.tail=TRUE, log.p=FALSE)  # CDF
ix <- seq(0,1,len=nAbsissa)
yi <- qhyper(ix, m=K, n=M-K, k=N, lower.tail=TRUE, log.p=FALSE) #iCDF
nSim <- 100
set.seed(41)
yran <- rhyper(nSim, m=K, n=M-K, k=N)

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('HypergeometricRV101R.eps')  # Start to save figure
RVdata1 <- data.frame(x,yp,yc)
RVdata2 <- data.frame(ix,yi)
RVdata3 <- data.frame(yran)
plot11 <- ggplot(RVdata1, aes(x,yp)) +
          geom_bar(stat="identity",fill="white",col="red") +
          geom_point(col="black",lwd=2) +
          xlab("x") +
          ggtitle("PDF")
plot12 <- ggplot(RVdata1, aes(x,yc)) +
          geom_step(col="green",lwd=1.2) +
          xlab("x") +
          ggtitle("CDF")
plot13 <- ggplot(RVdata2, aes(ix,yi)) +
          geom_step(col="blue",lwd=1.2) +
          xlab("x") + ylim(0,10) +
          ggtitle("Inverse CDF")
plot14 <- ggplot(RVdata3, aes(x=yran)) +
          geom_histogram(binwidth=0.25,fill="black",color="black")+
          xlab("x") +
          ggtitle("Histogram")
pushViewport(viewport(layout=grid.layout(2,2)))
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot13, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot14, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()            # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 초기하확률변수의 확률질량함수, 누적확률분포함수 그리고 역누적확률분포함수를 그린다. 또한, 이 확률분포에서 100개 초기하난수들을 생성하고, 이들의 히스토그램을 그린다. 이 그래프들이 그림 1.6.17에 그려져 있다. ■

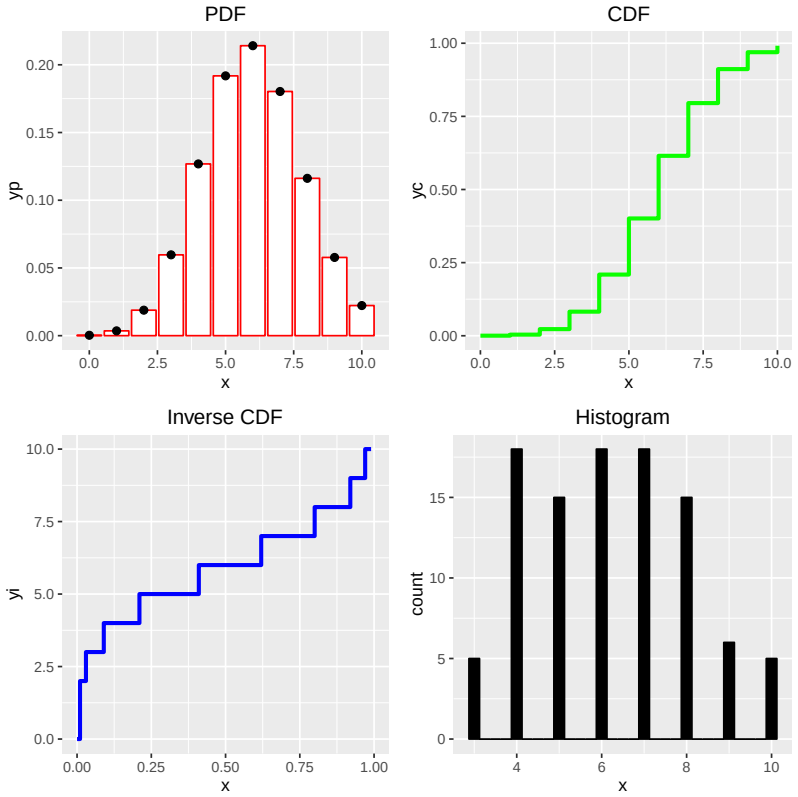


그림 1.6.17. 초기하확률변수와 R

1.6.9 다항확률분포

이항정리를 확장하면, 다음과 같다.

$$\left[\sum_{j=1}^{k} a_j\right]^n = \sum_{\sum_i x_i = n} \frac{n!}{\Pi_{i=1}^k x_i!} \Pi_{i=1}^k a_i^{x_i} \tag{1.6.21}$$

식 (1.6.21)을 이용해서, 다음과 같이 다항확률분포의 확률질량함수를 정의할 수 있다.

$$f(x_1, \cdots, x_k) = \frac{n!}{x_1! \cdots x_k!} p_1^{x_1} \cdots p_k^{x_k}, \quad \left(x_1 \geq 0, \cdots, x_k \geq 0, \sum_{i=1}^{k} x_i = n\right) \tag{1.6.22}$$

이 다항확률분포의 평균, 분산 그리고 공분산은 각각 다음과 같다.

$$E(x_i) = np_i, \quad Var(x_i) = np_i[1-p_i], \quad Cov(x_i, x_j) = -np_ip_j, \ (i \neq j) \tag{1.6.23}$$

MATLAB함수 mnpdf를 사용해서 다항확률질량함수를 계산할 수 있다.

예제 1.6.21 MATLAB을 이용해서 다항확률질량함수를 그리기 위해서, 다음 MATLAB

프로그램 MultinomialPMF101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: MultinomialPMF101.m
%  Multinomial Probability Mass Function
%------------------------------------------------------------------
clear all, close all
prob = [ 7 4 2 ]/13;              % Outcome probabilities
n = 12;                           % Sample size
[X1,X2] = meshgrid(0:n,0:n);
X3 = n-(X1+X2);
Y = mnpdf([X1(:),X2(:),X3(:)],repmat(prob,(n+1)^2,1));
% Plot the distribution
Y = reshape(Y,n+1,n+1);
bar3(Y)
set(gca,'fontsize',11,'fontweigh','bold')
set(gca,'XTickLabel',0:n,'YTickLabel',0:n)
xlabel('x_1'), ylabel('x_2')
zlabel('PMF')
saveas(gcf,'MultinomialPMF101','epsc')
save('MultinomialPMF101','Y')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 지지대가 $\{0, 1 \cdots, 12\}$이고 다음 식들을 만족하는 다항확률질량함수의 3D그래프를 출력한다.

$$p_1 = \frac{7}{13}, \quad p_2 = \frac{4}{13}, \quad p_3 = \frac{2}{13} \tag{1}$$

이 3D그래프가 그림 1.6.18에 그려져 있다. ■

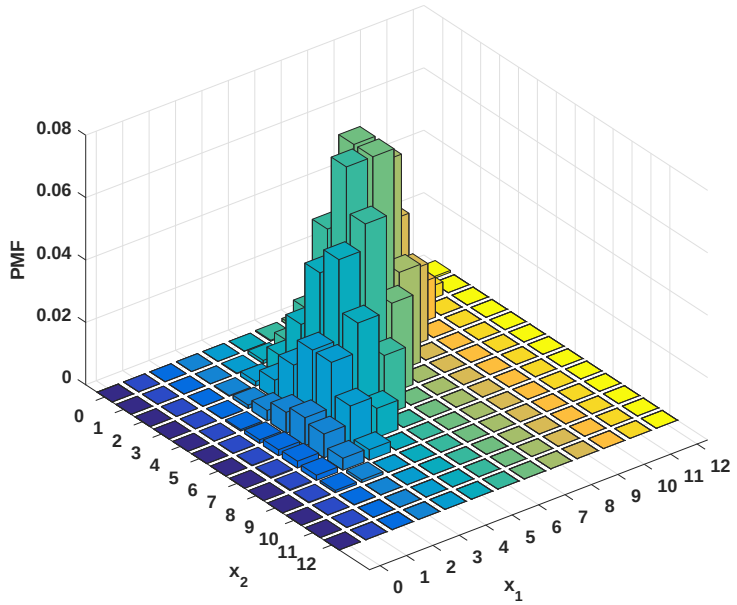


그림 1.6.18. 다항확률질량함수

예제 1.6.22 R을 이용해서 다항확률질량함수의 등고선도를 그리기 위해서 다음 R프로그램 MultinomialPMF101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: MultinomialPMF101R.R
# Multinomial Probabilitx12 Mass Function
# Programmed bx12 CBS
# ------------------------------------------------------------------------
p1 <- 7/14; p2 <- 4/14; p3 <- 2/13
n = 12
x1 <- seq(0,n,1)
x2 <- seq(0,n,1)
PM.ftn <- function(x1,x2){
            ifelse( x1+x2>n, 0,
                    gamma(n+1)/gamma(x1+1)/gamma(x2+1)/gamma(n-x1-x2+1)*
                    p1^x1*p2^x2*p3^(n-x1-x2) )
          }

# Contour Plot w/ ggplot2
# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
library(reshape2) # for melt()
setEPS()
plot.new()
postscript('MultinomialPMF101R.eps')  # Start to save figure
PM.lattice <- outer(x1,x2,PM.ftn)
PM.molten<-melt(PM.lattice)
names(PM.molten) <- c("x", "y", "z")
plot11 <- ggplot(PM.molten, aes(x,y,z=z)) +
          stat_contour(bins=6,aes(x,y,z=z), color="red", size=1.2)
plot12 <- ggplot(PM.molten, aes(x,y,z=z)) +
          geom_tile(aes(fill=z))+
          stat_contour(bins=6,aes(x,y,z=z), color="white", size=1.0)
plot21 <- ggplot(PM.molten, aes(x,y,z=z) )+
          geom_tile(aes(fill=z)) +
          stat_contour(bins=6,aes(x,y,z=z), color="black", size=0.9) +
          scale_fill_gradientn(colours=brewer.pal(6,"Greens"))
plot22 <- ggplot(PM.molten, aes(x,y,z=z)) +
          geom_tile(aes(fill=z)) +
          stat_contour(bins=6,aes(x,y,z=z), color="black", size=1.1) +
          scale_fill_gradientn(colours=brewer.pal(6,"YlOrRd"))
pushViewport(viewport(layout=grid.layout(2,2)))
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot21, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot22, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()             # End to save figure

# Contour Plot w/o ggplot2
# library(lattice) # for filled.contour()
# library(RColorBrewer) #for brewer.pal()
# PM.lattice <- outer(x1,x2,PM.ftn)
# filled.contour(x1,x2,PM.lattice,nlevels=5,col=brewer.pal(6,"YlOrRd"))
# ------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 지지대가 $\{0, 1 \cdots, 12\}$ 이고 다음 식들을 만족하는 다항확률질

량함수의 다양한 등고선도들을 출력한다.

$$p_1 = \frac{7}{13}, \quad p_2 = \frac{4}{13}, \quad p_3 = \frac{2}{13} \tag{1}$$

이 등고들이 그림 1.6.19에 그려져 있다. ■

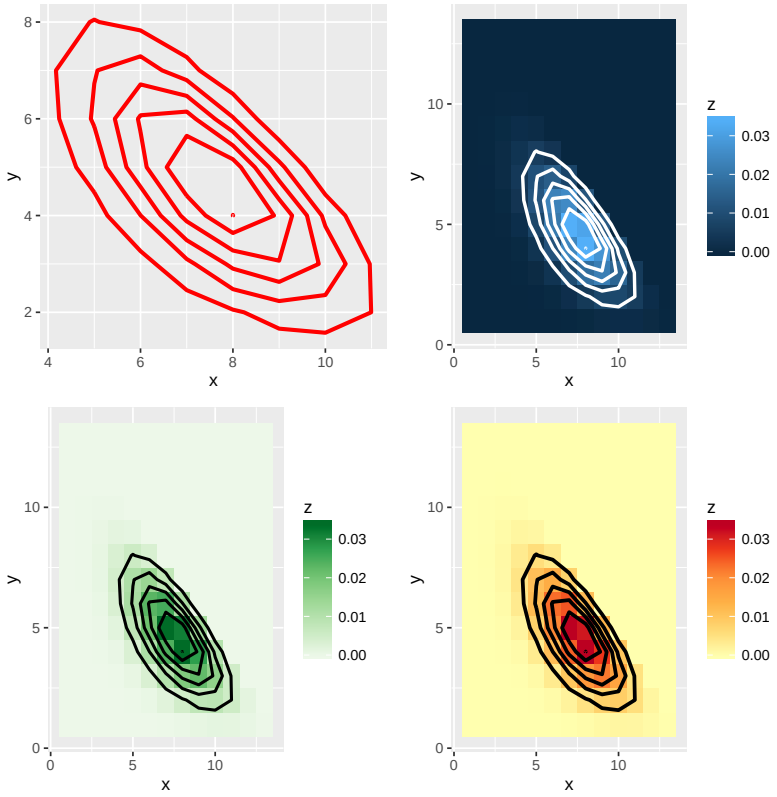


그림 1.6.19. 다항확률질량함수의 등고선도

MATLAB함수 mnrnd를 사용해서 다항난수들을 생성할 수 있다.

예제 1.6.23 MATLAB을 이용해서 다항난수를 생성하기 위해서 다음 MATLAB프로그램 MultinomialRV101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: MultinomialRV101.m
%  Multinomial Random Variable
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all
% Compute the distribution
prob = [ 7 4 2 ]/13;          % Outcome probabilities
n = 12;                       % Sample size
N = 100000;                   % Number of RV's
rand('twister',5489)
mRan = mnrnd(n,prob,N);
```

```
ih = zeros(n+1,n+1);
for i1 = 1:1:N
    ih(mRan(i1,1)+1,mRan(i1,2)+1) = ih(mRan(i1,1)+1,mRan(i1,2)+1)+1;
end
ih
% Plotting
bar3(ih')                   % Be careful for the Transpose of ih.
set(gca,'fontsize',11,'fontweigh','bold','XTickLabel',0:n, ...
    'YTickLabel',0:n)
xlabel('x_1'), ylabel('x_2')
zlabel('Histogram')
saveas(gcf,'MultinomialRV101','epsc')
save('MultinomialRV101','mRan')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 지지대가 $\{0, 1, \cdots, 12\}$이고 다음 식들을 만족하는 다항확률질량함수에 다항난수들을 출력한다.

$$p_1 = \frac{7}{13}, \quad p_2 = \frac{4}{13}, \quad p_3 = \frac{2}{13} \tag{1}$$

이 다항난수들 $\{[x_1, x_2]\}$의 산점도가 그림 1.6.20에 그려져 있다. ■

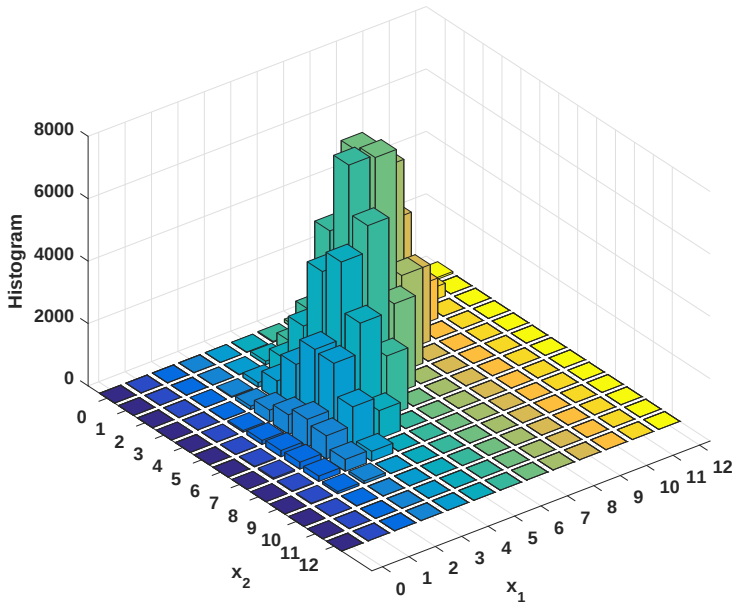


그림 1.6.20. 다항난수의 히스토그램

예제 1.6.24 R을 이용해서 다항난수를 생성하기 위해서 다음 R프로그램 MultinomialRV101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: MultinomialRV101R.R
```

```
# Multinomial Probability Mass Function
# Programmed by CBS
# --------------------------------------------------------------------------
p1 <- 7/14; p2 <- 4/14; p3 <- 2/13
n <- 12  # size
nSim <- 100
X <- rmultinom(nSim, size=n, prob=c(p1,p2,p3))

# Plotting
# install.packages("scatterplot3d")
library(scatterplot3d)
setEPS()
plot.new()
# postscript('MultinomialRV101R.eps')  # Start to save figure
Xdata <- data.frame(X)
attach(Xdata)
S3D <- scatterplot3d(X[1,],X[2,],X[3,],
                     pch=19,
                     xlab="x1",ylab="x2",zlab="x3",
                     type="p",
                     highlight.3d=TRUE)
FitPlane <- lm(X[3,] ~ X[1,]+X[2,])
S3D$plane3d(FitPlane)
#dev.off()            # End to save figure
# --------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 지지대가 $\{0, 1, \cdots, 12\}$이고 다음 식들을 만족하는 다항확률질량함수에 100개 다항난수들을 출력한다.

$$p_1 = \frac{7}{13}, \quad p_2 = \frac{4}{13}, \quad p_3 = \frac{2}{13} \tag{1}$$

이 다항난수들 $\{[x_1, x_2, x_3]\}$의 산점도가 그림 1.6.21에 그려져 있다. 그림 1.6.21에서 점의 색은 강도를 나타낸다. 즉, 몇 번 그 점이 출현했는가를 나타낸다. 또한, 이 점들이 다음 식을 만족함을 추가된 평면에서 알 수 있다.

$$x_1 + x_2 + x_3 = 12 \tag{2}$$

■

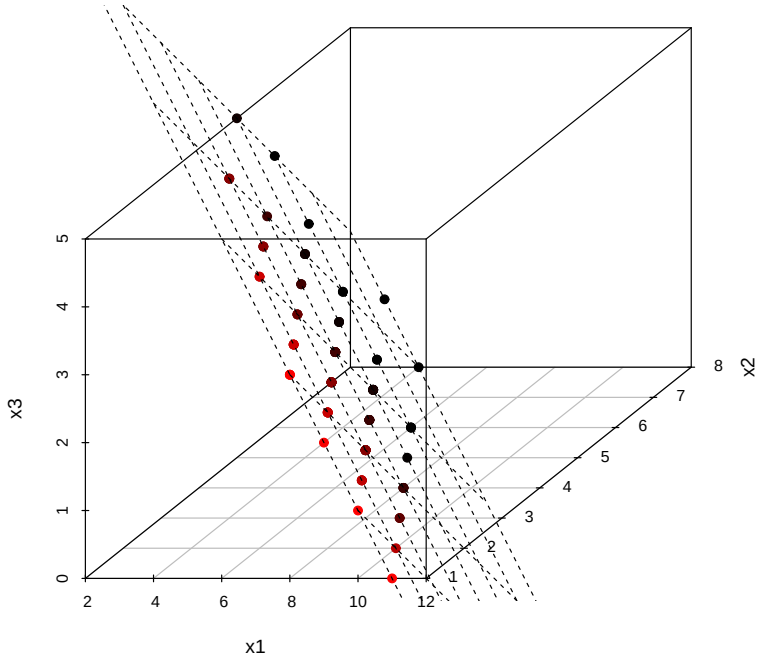


그림 1.6.21. 다항난수의 산점도

제 2 장

난수벡터

제2.1절 다변량확률분포

시뮬레이션을 깊게 이해하기 위해서는 다변량통계분석에 대해서 알고 있어야 한다. 이 절에서는 본서를 읽는데 필요한 선형대수학과 다변량정규확률분포의 특성에 대해 기술하고자 한다. 이에 대한 자세한 내용은 다변량통계분석의 고전인 Anderson [7]을 참조하라.

2.1.1 분할행렬

차원이 $m \times n$인 행렬 A를 다음과 같이 분할하자.

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \tag{2.1.1}$$

여기서 각 소행렬은 다음과 같다.

$$A_{11} \doteq [a_{ij}], \quad (i = 1, 2, \cdots, p, \quad j = 1, 2, \cdots, q) \tag{2.1.2}$$

$$A_{12} \doteq [a_{ij}], \quad (i = 1, 2, \cdots, p, \quad j = q+1, q+2, \cdots, n) \tag{2.1.3}$$

$$A_{21} \doteq [a_{ij}], \quad (i = p+1, p+2, \cdots, m, \quad j = 1, 2, \cdots, q) \tag{2.1.4}$$

$$A_{22} \doteq [a_{ij}], \quad (i = p+1, p+2, \cdots, m, \quad j = q+1, q+2, \cdots, n) \tag{2.1.5}$$

차원이 $m \times n$ 인 행렬 B 도 식 (2.1.1) 처럼 분해하면, 다음 식이 성립한다.

$$A + B = \begin{bmatrix} A_{11} + B_{11} & A_{12} + B_{12} \\ A_{21} + B_{21} & A_{22} + B_{22} \end{bmatrix} \tag{2.1.6}$$

차원이 $n \times l$ 인 행렬 C 를 다음과 같이 분해하자.

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \tag{2.1.7}$$

여기서 C_{11} 은 $q \times r$ 행렬, C_{12} 는 $q \times [l - r]$ 행렬, C_{21} 은 $[n - q] \times r$ 행렬, 그리고 C_{22} 는 $[n - q] \times [l - r]$ 행렬이다. 다음 식이 성립한다.

$$AC = \begin{bmatrix} A_{11}C_{11} + A_{12}C_{21} & A_{11}C_{12} + A_{12}C_{22} \\ A_{21}C_{11} + A_{22}C_{21} & A_{21}C_{12} + A_{22}C_{22} \end{bmatrix} \tag{2.1.8}$$

식 (2.1.8) 에서 알 수 있듯이, 차원이 $[m - p] \times q$ 인 행렬 X 에 대해서 다음 식이 성립한다.

$$\begin{bmatrix} I & O \\ X & I \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ XA_{11} + A_{21} & XA_{12} + A_{22} \end{bmatrix} \tag{2.1.9}$$

만약 A_{11} 과 A_{22} 가 정칙이면, 다음 행렬들을 정의하자.

$$A_{11 \cdot 2} \doteq A_{11} - A_{12}A_{22}^{-1}A_{21} \tag{2.1.10}$$

$$A_{22 \cdot 1} \doteq A_{22} - A_{21}A_{11}^{-1}A_{12} \tag{2.1.11}$$

식 (2.1.9) 에서 알 수 있듯이, 다음 식들이 성립한다.

$$\begin{bmatrix} I & O \\ -A_{21}A_{11}^{-1} & I \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ O & A_{22 \cdot 1} \end{bmatrix} \tag{2.1.12}$$

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} I & -A_{11}^{-1}A_{12} \\ O & I \end{bmatrix} = \begin{bmatrix} A_{11} & O \\ A_{21} & A_{22 \cdot 1} \end{bmatrix} \tag{2.1.13}$$

$$\begin{bmatrix} I & -A_{12}A_{22}^{-1} \\ O & I \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} A_{11\cdot 2} & O \\ A_{21} & A_{22} \end{bmatrix} \tag{2.1.14}$$

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} I & O \\ -A_{22}^{-1}A_{21} & I \end{bmatrix} = \begin{bmatrix} A_{11\cdot 2} & A_{12} \\ O & A_{22} \end{bmatrix} \tag{2.1.15}$$

식 (2.1.12) ∼ 식 (2.1.15)에서 알 수 있듯이, 다음 식들이 성립한다.

$$\begin{bmatrix} I & O \\ -A_{21}A_{11}^{-1} & I \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} I & -A_{11}^{-1}A_{12} \\ O & I \end{bmatrix} = \begin{bmatrix} A_{11} & O \\ O & A_{22\cdot 1} \end{bmatrix} \tag{2.1.16}$$

$$\begin{bmatrix} I & -A_{12}A_{22}^{-1} \\ O & I \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} I & O \\ -A_{22}^{-1}A_{21} & I \end{bmatrix} = \begin{bmatrix} A_{11\cdot 2} & O \\ O & A_{22} \end{bmatrix} \tag{2.1.17}$$

다음 식이 성립함은 자명하다.

$$\begin{bmatrix} I & O \\ A_{21} & I \end{bmatrix}^{-1} = \begin{bmatrix} I & O \\ -A_{21} & I \end{bmatrix} \tag{2.1.18}$$

식 (2.1.16) ∼ 식 (2.1.18)에서 알 수 있듯이, 다음 식들이 성립한다.

$$A = \begin{bmatrix} I & O \\ A_{21}A_{11}^{-1} & I \end{bmatrix} \begin{bmatrix} A_{11} & O \\ O & A_{22\cdot 1} \end{bmatrix} \begin{bmatrix} I & A_{11}^{-1}A_{12} \\ O & I \end{bmatrix} \tag{2.1.19}$$

$$A = \begin{bmatrix} I & A_{12}A_{11}^{-1} \\ O & I \end{bmatrix} \begin{bmatrix} A_{11\cdot 2} & O \\ O & A_{22} \end{bmatrix} \begin{bmatrix} I & O \\ A_{22}^{-1}A_{21} & I \end{bmatrix} \tag{2.1.20}$$

$$A = \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix} A_{11}^{-1} \begin{bmatrix} A_{11} & A_{12} \end{bmatrix} + \begin{bmatrix} O & O \\ O & A_{22\cdot 1} \end{bmatrix} \tag{2.1.21}$$

$$A = \begin{bmatrix} A_{12} \\ A_{22} \end{bmatrix} A_{22}^{-1} \begin{bmatrix} A_{21} & A_{22} \end{bmatrix} + \begin{bmatrix} A_{11\cdot 2} & O \\ O & O \end{bmatrix} \tag{2.1.22}$$

식 (2.1.18) ~ 식 (2.1.20)에서 알 수 있듯이, 다음 식들이 성립한다.

$$A^{-1} = \begin{bmatrix} I & -A_{11}^{-1}A_{12} \\ O & I \end{bmatrix} \begin{bmatrix} A_{11}^{-1} & O \\ O & A_{22\cdot 1}^{-1} \end{bmatrix} \begin{bmatrix} I & O \\ -A_{21}A_{11}^{-1} & I \end{bmatrix} \tag{2.1.23}$$

$$A^{-1} = \begin{bmatrix} A_{11}^{-1} + A_{11}^{-1}A_{12}A_{22\cdot 1}^{-1}A_{21}A_{11}^{-1} & -A_{11}^{-1}A_{12}A_{22\cdot 1}^{-1} \\ -A_{22\cdot 1}^{-1}A_{21}A_{11}^{-1} & A_{22\cdot 1}^{-1} \end{bmatrix} \tag{2.1.24}$$

$$A^{-1} = \begin{bmatrix} A_{11}^{-1} & O \\ O & O \end{bmatrix} + \begin{bmatrix} -A_{11}^{-1}A_{12} \\ I \end{bmatrix} A_{22\cdot 1}^{-1} \begin{bmatrix} -A_{21}A_{11}^{-1} & I \end{bmatrix} \tag{2.1.25}$$

$$A^{-1} = \begin{bmatrix} I & O \\ -A_{22}^{-1}A_{21} & I \end{bmatrix} \begin{bmatrix} A_{11\cdot 2}^{-1} & O \\ O & A_{22}^{-1} \end{bmatrix} \begin{bmatrix} I & -A_{21}A_{11}^{-1} \\ O & I \end{bmatrix} \tag{2.1.26}$$

$$A^{-1} = \begin{bmatrix} A_{11\cdot 2}^{-1} & -A_{11\cdot 2}^{-1}A_{12}A_{22}^{-1} \\ -A_{22}^{-1}A_{21}A_{11\cdot 2}^{-1} & A_{22}^{-1} + A_{22}^{-1}A_{21}A_{11\cdot 2}^{-1}A_{12}A_{22}^{-1} \end{bmatrix} \tag{2.1.27}$$

$$A^{-1} = \begin{bmatrix} O & O \\ O & A_{22}^{-1} \end{bmatrix} + \begin{bmatrix} I \\ -A_{22}^{-1}A_{21} \end{bmatrix} A_{11\cdot 2}^{-1} \begin{bmatrix} I & -A_{12}A_{22}^{-1} \end{bmatrix} \tag{2.1.28}$$

식 (2.1.26)에 식 (2.1.18)을 적용하면, 다음 식이 성립함을 알 수 있다.

$$\begin{bmatrix} A_{11\cdot 2}^{-1} & O \\ O & A_{22}^{-1} \end{bmatrix} = \begin{bmatrix} I & O \\ A_{22}^{-1}A_{21} & I \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}^{-1} \begin{bmatrix} I & A_{12}A_{22}^{-1} \\ O & I \end{bmatrix} \tag{2.1.29}$$

식 (2.1.29)에 식 (2.1.24)를 대입하면, 다음 식이 성립함을 알 수 있다.

$$A_{11\cdot 2}^{-1} = A_{11}^{-1} + A_{11}^{-1}A_{12}A_{22\cdot 1}^{-1}A_{21}A_{11}^{-1} \tag{2.1.30}$$

같은 방법으로 다음 식이 성립함을 알 수 있다.

$$A_{22\cdot 1}^{-1} = A_{22}^{-1} + A_{22}^{-1}A_{21}A_{11\cdot 2}^{-1}A_{12}A_{22}^{-1} \tag{2.1.31}$$

식 (2.1.30)과 식 (2.1.31)을 일반화하면, 적절한 차수를 갖는 행렬들 E, F, G 그리고 H에

대해서 다음 식들이 성립함을 알 수 있다.

$$[E+FGH]^{-1} = E^{-1} - E^{-1}F[G^{-1}+HE^{-1}F]^{-1}HE^{-1} \tag{2.1.32}$$

$$[E+FGH]^{-1} = E^{-1} - E^{-1}F[I+GHE^{-1}F]^{-1}GHE^{-1} \tag{2.1.33}$$

식 (2.1.10)과 식 (2.1.11)에서 알 수 있듯이, 다음 식이 성립한다.

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} I & -A_{11}^{-1}A_{12} \\ -A_{22}^{-1}A_{21} & I \end{bmatrix} \begin{bmatrix} A_{11\cdot 2}^{-1} & O \\ O & A_{22\cdot 1}^{-1} \end{bmatrix} = I \tag{2.1.34}$$

즉, 다음 식이 성립한다.

$$A^{-1} = \begin{bmatrix} I & -A_{11}^{-1}A_{12} \\ -A_{22}^{-1}A_{21} & I \end{bmatrix} \begin{bmatrix} A_{11\cdot 2}^{-1} & O \\ O & A_{22\cdot 1}^{-1} \end{bmatrix} \tag{2.1.35}$$

같은 방법으로 다음 식을 증명할 수 있다.

$$A^{-1} = \begin{bmatrix} A_{11\cdot 2}^{-1} & O \\ O & A_{22\cdot 1}^{-1} \end{bmatrix} \begin{bmatrix} I & -A_{12}A_{22}^{-1} \\ -A_{21}A_{11}^{-1} & I \end{bmatrix} \tag{2.1.36}$$

다음 식이 성립함을 쉽게 알 수 있다.

$$\begin{vmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{vmatrix} = |A_{11}||A_{22}| \tag{2.1.37}$$

식 (2.1.16), 식 (2.1.17)과 식 (2.1.37)에서 알 수 있듯이, 다음 식들이 성립한다.

$$|A| = |A_{11}| \cdot |A_{22} - A_{21}A_{11}^{-1}A_{12}| = |A_{11}| \cdot |A_{22\cdot 1}| \tag{2.1.38}$$

$$|A| = |A_{22}| \cdot |A_{11} - A_{12}A_{22}^{-1}A_{21}| = |A_{22}| \cdot |A_{11\cdot 2}| \tag{2.1.39}$$

2.1.2 행렬의 분해

[1] LU분해

LU분해는 Gauss소거법을 행렬로 표현한 것이다. 만약 $n \times n$ 행렬 $A = [a_{ij}]$, 하삼각행렬

$L = [l_{ij}]$과 상삼각행렬 $U = [u_{ij}]$이 다음 식을 만족하면, 행렬 A가 LU분해되었다고 한다.

$$A = LU \tag{2.1.40}$$

행렬 A의 원소들은 n^2이고 행렬 L과 행렬 U는 각각 $n[n+1]/2$개 원소들을 포함하고 있다. 따라서, 식 (2.1.40)의 LU분해는 일의적이 아니다. 따라서, 식 (2.1.40)의 우변에 제약을 가해야 한다. 만약 행렬 L의 대각원소들이 모두 1이면, LU분해를 Doolittle분해라 부른다. 만약 행렬 U의 대각원소들이 모두 1이면, LU분해를 Crout분해라 부른다.

알고리즘 2.1.1: Doolittle분해

(1단계) 각 $i(=1,2,\cdots,n)$에 대해서, $l_{ii}=1$이라 놓는다.

(2단계) 각 $j(=1,2,\cdots,n)$에 대해서, 다음과 같은 과정을 수행한다.

(2-1) 각 $i\,(=1,2,\cdots,j)$에 대해서, 다음과 같이 u_{ij}를 계산한다.

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik}u_{kj}$$

(2-2) 각 $i\,(=j+1,j+2,\cdots,n)$에 대해서, 다음과 같이 l_{ij}를 계산한다.

$$l_{ij} = \frac{1}{u_{jj}}\left[a_{ij} - \sum_{k=1}^{i-1} l_{ik}u_{kj}\right]$$

알고리즘 2.1.1에서 알 수 있듯이, $\prod_{j=1}^{n} u_{jj}$가 0이 아닌 경우에 한해서 LU분해를 할 수 있다. 만약 행렬 A가 대칭행렬이면, 식 (2.1.40)의 LU분해를 다음과 같이 쓸 수 있다.

$$A = LL^t \tag{2.1.41}$$

식 (2.1.41)을 Cholesky분해라고 부른다. 행렬 L의 각 대각원소가 양수라고 가정하면, Cholesky분해는 일의적으로 정해진다.

알고리즘 2.1.2: Cholesky분해

(1단계) 다음 값들을 계산한다.

$$l_{11} = \sqrt{a_{11}}, \quad l_{i1} = \frac{a_{i1}}{\sqrt{a_{11}}}, \quad (i = 2, 3, \cdots, n)$$

(2단계) 각 $j(= 2, 3, \cdots, n)$에 대해서, 다음 계산을 한다.

$$l_{jj} = \sqrt{a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2}$$
$$l_{ij} = \frac{1}{l_{jj}} \left[a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk} \right], \quad (i = j+1, j+2, \cdots, n-1)$$

예제 2.1.1 MATLAB을 이용해서 LU분해와 Cholesky분해를 하는 예로서, 다음 MATLAB 프로그램 LUdecomposition101.m을 실행해 보자.

```
% ------------------------------------------------------------------
% Filename: LUdecomposition101.m
% LU decomposition + Cholesky decomposition
% ------------------------------------------------------------------
clear all, close all
diary LUdecomposition101.txt
L = [ 1 0; 2 3 ]
A = L*L'
[ L1 U1 P1 ] = lu(A)
D1 = P1*A - L1*U1
[ L2 U2 ] = lu(A)
D2 = A - L2*U2
U3 = chol(A)
L3 = chol(A,'lower')
D3 = A - L3*L3'
diary off
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램에서 정의한 행렬 L과 행렬 A는 각각 다음과 같다.

$$L = \begin{bmatrix} 1 & 0 \\ 2 & 3 \end{bmatrix}, \quad A = \begin{bmatrix} 1 & 2 \\ 2 & 13 \end{bmatrix} \tag{1}$$

첫 번째 MATLAB lu명령문 `'[ L1 U1 P1 ] = lu(A)'`는 LU분해 $P_1 A = L_1 U_1$을 실행하기 위한 것이다. 여기서 P_1은 순열행렬(permutation matrix)이다. 이 명령문을 실행한

결과는 다음과 같다.

$$L_1 = \begin{bmatrix} 1 & 0 \\ 1/2 & 1 \end{bmatrix}, \quad U_1 = \begin{bmatrix} 2 & 13 \\ 0 & -4.5 \end{bmatrix}, \quad P_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{2}$$

두 번째 MATLAB lu명령문 `'[ L2 U2 ] = lu(A)'`는 LU분해 $A = L_2U_2$을 실행하기 위한 것이다. 이 명령문을 실행한 결과는 다음과 같다.

$$L_2 = \begin{bmatrix} 1/2 & 1 \\ 1 & 0 \end{bmatrix}, \quad U_1 = \begin{bmatrix} 2 & 13 \\ 0 & -4.5 \end{bmatrix} \tag{3}$$

첫 번째 MATLAB chol명령문 `'U3 = chol(A)'`는 Cholesky분해 $A = U_3^tU_3$를 실행하기 위한 것이다. 이 명령문을 실행한 결과는 다음과 같다.

$$U_3 = \begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix} \tag{4}$$

두 번째 MATLAB chol명령문 `'U3 = chol(A,'lower')'`는 Cholesky분해 $A = L_3L_3^t$를 실행하기 위한 것이다. 이 명령문을 실행한 결과는 다음과 같다.

$$L_3 = \begin{bmatrix} 1 & 0 \\ 2 & 3 \end{bmatrix} \tag{5}$$

■

예제 2.1.2 R을 이용해서 LU분해와 Cholesky분해를 하는 예로서, 다음 R프로그램 LUdecomposition101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: LUdecomposition101R.R
# LU Decomposition
# Programmed by CBS
# ------------------------------------------------------------------------
# install.packages("Matrix")
library(Matrix)
# Making a matrix
L <- matrix( c(1,0,2,3), nrow=2, byrow=TRUE)
( A <- L%*%t(L) )

# LU decomposition
( luA <- lu(A) )
```

```
( EluA <- expand(luA) )
( L <- EluA$L )
( U <- EluA$U )
( P <- EluA$P )
( dum1 <- A - P%*%L%*%U )

# Cholesky decomposition
( CholA <- chol(A) )
print(CholA)
print(t(CholA)%*%CholA-A)
# ------------------------------------------------------------------------------
```

이 R프로그램에서 정의한 행렬 L과 행렬 A는 각각 다음과 같다.

$$L = \begin{bmatrix} 1 & 0 \\ 2 & 3 \end{bmatrix}, \quad A = \begin{bmatrix} 1 & 2 \\ 2 & 13 \end{bmatrix} \tag{1}$$

R패키지 Matrix의 lu분해함수인 lu를 사용한 명령문 `'luA <- lu(A)'`를 실행하면, 행렬 A의 LU분해 $A = LU$가 실행된다. 이 R함수 lu는 원래 성긴(sparse) 행렬을 압축시켜서 LU분해를 하므로, 압축된 결과를 풀어야 원하는 lu분해를 얻을 수 있다. 이러한 과정을 수행하기 위해서 명령문 `'EluA <- expand(luA)'`를 실행하면, 다음 결과를 얻는다.

$$L = \begin{bmatrix} 1 & 0 \\ 0.5 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 2 & 13 \\ 0 & -4.5 \end{bmatrix}, \quad P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{2}$$

R의 base에 포함된 Cholesky분해함수인 chol을 사용한 명령문 `'U3 = chol(A)'`를 실행하면, 행렬 A의 Cholesky분해 $A = U^tU$가 실행된다. 이 명령문을 실행한 결과는 다음과 같다.

$$U = \begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix} \tag{3}$$

■

[2] Schur분해

차원이 $n \times n$인 정방행렬 A에 대해서, 다음 식을 만족하는 스칼라 λ를 A의 고유값(eigenvalue)이라고 하고 $\boldsymbol{x}(\neq \mathbf{0})$를 고유값 λ에 대한 고유벡터(eigenvector)라고 한다.

$$A\boldsymbol{x} = \lambda\boldsymbol{x} \tag{2.1.42}$$

다음과 같이 행렬 A의 고유멱함수(characteristic polynomial)를 정의하자.

$$f_A(z) \doteq |zI_n - A| \doteq z^n + a_{n-1}z^{n-1} + a_{n-2}z^{n-2} + \cdots + a_1 z + a_0 \tag{2.1.43}$$

식 (2.1.42)에서 알 수 있듯이, 고유값 λ는 다음 고유방정식(characteristic equation)을 만족한다.

$$f_A(\lambda) = 0 \tag{2.1.44}$$

행렬 A의 고유값들 $\lambda_1, \lambda_2, \cdots, \lambda_n$에 대응하는 고유벡터들 $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n$이 서로 일차독립이라고 가정하고, 행렬 A에 대해 적당한 정칙행렬 P를 찾아서 $P^{-1}AP$가 대각행렬이 되게 할 수 있다. 이런 경우에 행렬 A는 대각행렬과 유사하다고(similar) 한다. 다음과 같이 차원이 $n \times n$인 행렬 P를 정의하자.

$$P \doteq [\boldsymbol{x}_1\, \boldsymbol{x}_2\, \cdots\, \boldsymbol{x}_n] \tag{2.1.45}$$

다음 식들이 성립한다.

$$AP = [A\boldsymbol{x}_1\, A\boldsymbol{x}_2\, \cdots, A\boldsymbol{x}_n] = [\lambda_1\boldsymbol{x}_1\, \lambda_2\boldsymbol{x}_2\, \cdots\, \lambda_n\boldsymbol{x}_n] \tag{2.1.46}$$

즉, 다음 식이 성립한다.

$$AP = [\boldsymbol{x}_1\, \boldsymbol{x}_2\, \cdots\, \boldsymbol{x}_{n-1}\, \boldsymbol{x}_n] \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 & 0 \\ 0 & \lambda_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \lambda_{n-1} & 0 \\ 0 & 0 & \cdots & 0 & \lambda_n \end{bmatrix} \tag{2.1.47}$$

여기서 대각행렬 Λ는 다음과 같다.

$$\Lambda \doteq diag(\lambda_1, \lambda_2, \cdots, \lambda_{n-1}, \lambda_n) \doteq \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 & 0 \\ 0 & \lambda_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & \lambda_{n-1} & 0 \\ 0 & 0 & \cdots & 0 & \lambda_n \end{bmatrix} \tag{2.1.48}$$

가정에 의해서 행렬 P는 정칙이므로, P의 역행렬 P^{-1}가 존재한다. 따라서, 식 (2.1.48)을 다음과 같이 쓸 수 있다.

$$P^{-1}AP = \Lambda \tag{2.1.49}$$

식 (2.1.49)는 행렬 A를 대각화하는(diagonalization) 것이다. 식 (2.1.49)를 다음과 같이 쓸 수 있다.

$$A = P\Lambda P^{-1} \tag{2.1.50}$$

행렬 A의 서로 다른 고유값들에 대한 고유벡터들은 일차독립이다. 따라서, 행렬 A의 고유값들이 서로 다르다면, 행렬 A를 대각화할 수 있다. 또한, 대칭행렬은 대각화할 수 있음을 증명할 수 있다. 그러나, 모든 행렬을 대각화할 수는 없다. 대각화가 되지 않는 경우에는 Jordan형 행렬을 사용한다.

만약 행렬 A가 대칭이면, 다음 식들을 만족하는 행렬 Q가 존재한다.

$$A = Q\Lambda Q^t, \quad Q^tQ = I \tag{2.1.51}$$

행렬 Q는 직교행렬이다.

식 (2.1.51)의 분해는 다음과 같은 Schur분해의 특수한 경우이다. 행렬 Q의 각 원소의 켤레복소수로 이루어진 행렬을 $\bar{Q}$로 표기하고, 행렬 $\bar{Q}^t$를 Q^*로 표기하자. 식 $QQ^* = I$를 만족하는 정방행렬 Q를 유니타리행렬(unitary matrix)이라 한다. 정방행렬 A에 대해서, 다음 식을 만족하는 삼각행렬 T와 유니타리행렬 Q가 존재한다.

$$A = QTQ^* \tag{2.1.52}$$

여기서 T는 대각원소들이 행렬 A의 고유값들로 이루어진 상삼각행렬이다.

예제 2.1.3 MATLAB을 이용해서 Schur분해를 하는 예로서, 다음 MATLAB프로그램 SchurDecomposition101.m을 실행해 보자.

```
% --------------------------------------------------------------------------
% Filename: SchurDecomposition101.m
% Schur Dcomposition
% --------------------------------------------------------------------------
clear all, close all
diary SchurDecomposition101.txt
T = [ 1 2 ; 0 3 ]
Q = 1/sqrt(2)*[ 1 1 ; 1 -1 ]
A = Q*T*Q'
T1 = schur(A)
[ Q2 T2 ] = schur(A)
D2 = A - Q2*T2*Q2'
diary off
% End of program
% --------------------------------------------------------------------------
```

이 MATLAB프로그램에서 정의한 행렬들 T와 Q, 그리고 행렬 A는 다음과 같다.

$$T = \begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix}, \quad Q = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad A = \begin{bmatrix} 3 & -2 \\ 0 & 1 \end{bmatrix} \tag{1}$$

첫 번째 MATLAB schur명령문 `'T1 = schur(A)'`를 실행한 결과는 다음과 같다.

$$T = \begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix} \tag{2}$$

두 번째 MATLAB schur명령문 `'[ Q2 T2 ] = schur(A)'`를 실행한 결과는 다음과 같다.

$$T_2 = \begin{bmatrix} 3 & -2 \\ 0 & 1 \end{bmatrix}, \quad Q_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3}$$

■

예제 2.1.4 R을 이용해서 Schur분해를 하는 예로서, 다음 R프로그램 SchurDecomposition101R.R을 실행해 보자.

```
# --------------------------------------------------------------------------
# Filename: SchurDecomposition101R.R
# Schur Decomposition
```

```
# Programmed by CBS
# ------------------------------------------------------------------------
# install.packages("Matrix")
library(Matrix)
# Making a matrix
T <- matrix( c(1,2,0,3), nrow=2, byrow=TRUE)
Q <- 1/sqrt(2)*matrix( c(1,1,1,-1), nrow=2, byrow=TRUE)
( A <- Q%*%T%*%t(Q) )

# Schur decomposition
( T1 <- Schur(A) )
( Q2 <- T1$Q )
( T2 <- T1$T )
print(Q2%*%T2%*%t(Q2)-A)
# ------------------------------------------------------------------------
```

이 R프로그램에서 정의한 행렬들 T와 Q, 그리고 행렬 A는 다음과 같다.

$$T = \begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix}, \quad Q = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad A = \begin{bmatrix} 3 & -2 \\ 0 & 1 \end{bmatrix} \tag{1}$$

R schur명령문 `'T1 = schur(A)'`를 실행한 결과는 다음과 같다.

$$T_2 = \begin{bmatrix} 3 & -2 \\ 0 & 1 \end{bmatrix}, \quad Q_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2}$$

■

[3] 특이값분해

선형대수학에서 특이값분해(singular value decomposition: SVD)는 상당히 중요한 위치를 차지한다. 특이값분해에 대한 자세한 내용은 최병선 [1, pp. 39]을 참조하라.

어떤 $n \times p$행렬 A에 대해서, 다음 식을 만족하는 $n \times n$직교행렬 P, $p \times p$직교행렬 Q 그리고 $n \times p$행렬 D가 존재한다. 여기서 $n \geq p$이다.

$$A = PDQ \tag{2.1.53}$$

만일 행렬 A의 랭크가 $r(\leq p)$이면, 행렬 D는 다음과 같은 형태를 갖는다.

$$D = \begin{bmatrix} D_1 & O \\ O & O \end{bmatrix} \tag{2.1.54}$$

여기서 $r \times r$ 대각행렬 D_1 의 각 대각원소는 양수이다.

예제 2.1.5 MATLAB을 이용해서 특이값분해를 하는 예로서, 다음 MATLAB프로그램 SVDdecomposition101.m을 실행해 보자.

```
% ------------------------------------------------------------------
% Filename: SVDdecomposition101.m
% Singular Value Decomposition
% ------------------------------------------------------------------
clear all, close all
diary SVDdecomposition101.txt
A = magic(3)
[ P1 D1 Q1 ] = svd(A)
Diff1 = A - P1*D1*Q1'
D2 = svd(A)
diary off
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램에서 정의한 행렬 A는 다음과 같다.

$$A = \begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix} \tag{1}$$

첫 번째 MATLAB svd명령문 `'[ P1 D1 Q1 ] = svd(A)'`를 실행한 결과는 다음과 같다.

$$P_1 = \begin{bmatrix} -0.5774 & 0.7071 & 0.4082 \\ -0.5774 & 0.0000 & -0.8165 \\ -0.5774 & -0.7071 & 0.4082 \end{bmatrix} \tag{2}$$

$$D_1 = \begin{bmatrix} 15.0000 & 0 & 0 \\ 0 & 6.9282 & 0 \\ 0 & 0 & 3.4641 \end{bmatrix} \tag{3}$$

$$Q_1 = \begin{bmatrix} -0.5774 & 0.4082 & 0.7071 \\ -0.5774 & -0.8165 & -0.0000 \\ -0.5774 & 0.4082 & -0.7071 \end{bmatrix} \tag{4}$$

두 번째 MATLAB svd명령문 'D2 = svd(A)'를 실행한 결과는 다음과 같다.

$$D_2 = \begin{bmatrix} 15.0000 \\ 6.9282 \\ 3.4641 \end{bmatrix} \tag{5}$$

■

예제 2.1.6 R을 이용해서 특이값분해를 하는 예로서, 다음 R프로그램 SVDdecomposition101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: SVDdecomposition101R.R
# SVD Decomposition
# Programmed by CBS
# ------------------------------------------------------------------------
A <- matrix( c(8,1,6,3,5,7,4,9,2), nrow=3, byrow=TRUE)
( SVDD <- svd(A) )
( U <- SVDD$u )
( V <- SVDD$v )
( D <- SVDD$d )
( Diagg <- diag(SVDD$d) )
( dum1 <- A - U%*%Diagg%*%t(V) )  # Check  A = UDV'
( dum2 <- Diagg - t(U)%*%A%*%V )  # Check  D = U'AV
# ------------------------------------------------------------------------
```

이 R프로그램에서 정의한 행렬 A는 다음과 같다.

$$A = \begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix} \tag{1}$$

R svd명령문 'SVDD <- svd(A)' 를 실행한 결과는 다음과 같다.

$$U = \begin{bmatrix} -0.5774 & 0.7071 & 0.4082 \\ -0.5774 & 0.0000 & -0.8165 \\ -0.5774 & -0.7071 & 0.4082 \end{bmatrix} \tag{2}$$

$$D = \begin{bmatrix} 15.0000 & 0 & 0 \\ 0 & 6.9282 & 0 \\ 0 & 0 & 3.4641 \end{bmatrix} \tag{3}$$

$$V = \begin{bmatrix} -0.5774 & 0.4082 & 0.7071 \\ -0.5774 & -0.8165 & -0.0000 \\ -0.5774 & 0.4082 & -0.7071 \end{bmatrix} \tag{4}$$

이 행렬들은 다음 식들을 만족한다.

$$UDV^t = A, \quad U^tAV = D \tag{5}$$

■

2.1.3 다변량정규확률분포

확률벡터 $\boldsymbol{x} = [x_1, x_2, \cdots, x_p]^t$ 의 평균벡터 $\boldsymbol{\mu} = [\mu_1, \mu_2, \cdots, \mu_p]^t$ 이고 분산공분산행렬이 Σ 라 하자. 만약 임의의 p차원 실벡터 (real vector) $\boldsymbol{a}$에 대해서 $\boldsymbol{a}^t\boldsymbol{x}$가 정규확률분포를 따르면, 확률벡터 $\boldsymbol{x}$는 p변량 정규확률분포를 따른다고 하고, 이를 다음과 같이 표기하자.

$$\boldsymbol{x} \overset{d}{\sim} \mathcal{N}_p(\boldsymbol{\mu}, \Sigma) \tag{2.1.55}$$

또한, $\mathcal{N}_p$ 대신 $\mathcal{N}$을 사용하기도 한다. 본서에서는 분산공분산행렬 Σ가 양정치행렬이라고 가정하자.

이 p변량 정규확률분포의 결합확률밀도함수는 다음과 같다.

$$n(\boldsymbol{x} \mid \mu, \Sigma) \doteq \frac{1}{\sqrt{[2\pi]^p|\Sigma|}} \exp\left(-\frac{1}{2}[\boldsymbol{x} - \boldsymbol{\mu}]^t\Sigma^{-1}[\boldsymbol{x} - \boldsymbol{\mu}]\right) \tag{2.1.56}$$

이 $n(\boldsymbol{x} \mid \mu, \Sigma)$가 결합확률밀도함수임을 증명하기 위해서는 보통 p차원 극좌표 (polar coor-

dinates)를 사용한다. 이 방법은 Schur분해를 사용하는 특별한 경우이므로, 여기서는 Schur 분해를 사용해서 $n(\boldsymbol{x} \mid \mu, \Sigma)$가 결합확률밀도함수임을 증명하기로 하자. 행렬 Σ가 양정치이므로, 다음 식을 만족하는 행렬 Q가 존재한다.

$$\Sigma = Q\Lambda Q^t, \quad Q^tQ = I \tag{2.1.57}$$

식 (2.1.51)에서 알 수 있듯이, 대각행렬 Λ의 대각원소인 각 고유값 λ_i는 양수이다. 다음과 같이 행렬들 $\Lambda^{1/2}$와 $\Lambda^{-1/2}$를 정의하자.

$$\Lambda^{1/2} \doteq diag(\lambda_1^{1/2}, \lambda_2^{1/2}, \cdots, \lambda_p^{1/2}) \tag{2.1.58}$$

$$\Lambda^{-1/2} \doteq diag(\lambda_1^{-1/2}, \lambda_2^{-1/2}, \cdots, \lambda_p^{-1/2}) \tag{2.1.59}$$

또한, 행렬들 $\Sigma^{1/2}$와 $\Sigma^{-1/2}$를 각각 다음과 같이 정의하자.

$$\Sigma^{1/2} = Q\Lambda^{1/2}Q^t, \quad \Sigma^{-1/2} = Q\Lambda^{-1/2}Q^t \tag{2.1.60}$$

행렬 Q가 직교행렬이므로, 행렬식 $|Q|$는 ± 1이다. 다음과 같은 변수변환을 생각해보자.

$$\boldsymbol{z} = [z_1, z_2, \cdots, z_p]^t \doteq \Sigma^{-1/2}[\boldsymbol{x} - \boldsymbol{\mu}] \tag{2.1.61}$$

이 변수변환의 Jacobian J는 다음과 같다.

$$J = \left| \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}} \right| = |\Sigma|^{1/2} \tag{2.1.62}$$

따라서, 다음 식들이 성립한다.

$$\begin{aligned} &\int \frac{1}{\sqrt{[2\pi]^p |\Sigma|}} \exp\left(-\frac{1}{2}[\boldsymbol{x} - \boldsymbol{\mu}]^t \Sigma^{-1} [\boldsymbol{x} - \boldsymbol{\mu}] \right) d\boldsymbol{x} \\ &= \int \frac{1}{\sqrt{[2\pi]^p}} \exp\left(-\frac{1}{2} \boldsymbol{z}^t \boldsymbol{z} \right) d\boldsymbol{z} \end{aligned} \tag{2.1.63}$$

식 (2.1.63)의 우변을 다음과 같이 쓸 수 있다.

$$\prod_{j=1}^{p} \int \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2} z_i^2 \right) dz_i = \prod_{i=1}^{p} 1 = 1 \tag{2.1.64}$$

즉, 다음 식이 성립한다.

$$\int n(\boldsymbol{x} \mid \mu, \Sigma) d\boldsymbol{x} = 1 \tag{2.1.65}$$

따라서 $n(\boldsymbol{x} \mid \mu, \Sigma)$는 결합확률밀도함수이다.

다음 식들이 성립한다.

$$\int \boldsymbol{x}\, n(\boldsymbol{x} \mid \boldsymbol{\mu}, \Sigma) d\boldsymbol{x} = \int [\boldsymbol{x} - \boldsymbol{\mu}]\, n(\boldsymbol{x} \mid \boldsymbol{\mu}, \Sigma) d\boldsymbol{x} + \int \boldsymbol{\mu}\, n(\boldsymbol{x} \mid \boldsymbol{\mu}, \Sigma) d\boldsymbol{x} \tag{2.1.66}$$

변수변환 (2.1.61)을 적용하면, 식 (2.1.66)의 우변을 다음과 같이 쓸 수 있다.

$$\begin{aligned} &\Sigma^{1/2} \int \boldsymbol{z} n(\boldsymbol{z} \mid \boldsymbol{0}, I) d\boldsymbol{x} + \boldsymbol{\mu} \int n(\boldsymbol{x} \mid \boldsymbol{\mu}, \Sigma) d\boldsymbol{x} \\ &= \Sigma^{1/2} \prod_{i=1}^{p} \int z_i \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2} z_i^2\right) dz_i + \boldsymbol{\mu} = \boldsymbol{\mu} \end{aligned} \tag{2.1.67}$$

여기서 두 번째 등호는 식 (2.1.65)에 의해서 성립한다. 변수변환 (2.1.61)을 적용하면, 다음 식이 성립함을 알 수 있다.

$$\int [\boldsymbol{x} - \boldsymbol{\mu}][\boldsymbol{x} - \boldsymbol{\mu}]^t n(\boldsymbol{x} \mid \boldsymbol{\mu}, \Sigma) d\boldsymbol{x} = \int \Sigma^{1/2} \boldsymbol{z}\boldsymbol{z}^t \Sigma^{1/2} n(\boldsymbol{z} \mid \boldsymbol{0}, I) d\boldsymbol{z} \tag{2.1.68}$$

식 (2.1.68)의 우변을 다음과 같이 쓸 수 있다.

$$\begin{aligned} &\Sigma^{1/2} \int \boldsymbol{z}\boldsymbol{z}^t n(\boldsymbol{z} | \boldsymbol{0}, I) d\boldsymbol{z} \Sigma^{1/2} \\ &= \prod_{i=1}^{p} \int z_i^2 \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2} z_i^2\right) dz_i \Sigma = \Sigma \prod_{i=1}^{p} 1 = \Sigma \end{aligned} \tag{2.1.69}$$

식 (2.1.67)과 식 (2.1.69)에서 알 수 있듯이, 다음 식들이 성립한다.

$$E(\boldsymbol{x}) = \int [\boldsymbol{x} - \boldsymbol{\mu}] n(\boldsymbol{x} \mid \boldsymbol{\mu}, \Sigma) d\boldsymbol{x} = \boldsymbol{\mu} \tag{2.1.70}$$

$$Var(\boldsymbol{x}) = \int [\boldsymbol{x} - \boldsymbol{\mu}][\boldsymbol{x} - \boldsymbol{\mu}]^t n(\boldsymbol{x} \mid \boldsymbol{\mu}, \Sigma) d\boldsymbol{x} = \Sigma \tag{2.1.71}$$

식 (2.1.65)에서 알 수 있듯이, 평균벡터가 $\boldsymbol{\mu}$이고 분산공분산행렬이 Σ인 다변량정규확률

분포에서 다음 식이 성립한다.

$$\int \exp\left(-\frac{1}{2}[\boldsymbol{x}-\boldsymbol{\mu}]^t\Sigma^{-1}[\boldsymbol{x}-\boldsymbol{\mu}]\right)d\boldsymbol{x} = \sqrt{[2\pi]^p|\Sigma|} \tag{2.1.72}$$

따라서, 다음 식이 성립한다.

$$\int \exp\left(-\frac{1}{2}\boldsymbol{x}^t\Sigma^{-1}\boldsymbol{x}+\boldsymbol{\mu}^t\Sigma^{-1}\boldsymbol{x}\right)d\boldsymbol{x} = \sqrt{[2\pi]^p|\Sigma|}\exp\left(\frac{1}{2}\boldsymbol{\mu}^t\Sigma^{-1}\boldsymbol{\mu}\right) \tag{2.1.73}$$

식 (2.1.73) 에 식 $\boldsymbol{\mu}=\Sigma\boldsymbol{g}$에 대입하면, 다음 식을 얻는다.

$$\int \exp\left(-\frac{1}{2}\boldsymbol{x}^t\Sigma^{-1}\boldsymbol{x}+\boldsymbol{g}^t\boldsymbol{x}\right)d\boldsymbol{x} = \sqrt{[2\pi]^p|\Sigma|}\exp\left(\frac{1}{2}\boldsymbol{g}^t\Sigma\boldsymbol{g}\right) \tag{2.1.74}$$

다변량정규확률분포 $\mathcal{N}(\boldsymbol{\mu},\Sigma)$를 따르는 확률벡터 $\boldsymbol{x}$, 평균벡터 $\boldsymbol{\mu}$, 그리고 분산공분산행렬 Σ를 다음과 같이 분할하기로 하자.

$$\boldsymbol{x}=\begin{bmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{bmatrix},\quad \boldsymbol{\mu}=\begin{bmatrix}\boldsymbol{\mu}_1\\ \boldsymbol{\mu}_2\end{bmatrix},\quad \Sigma=\begin{bmatrix}\Sigma_{11} & \Sigma_{12}\\ \Sigma_{21} & \Sigma_{22}\end{bmatrix} \tag{2.1.75}$$

식 (2.1.25)에서 알 수 있듯이, 다음 식들이 성립한다.

$$\begin{aligned}&[\boldsymbol{x}-\boldsymbol{\mu}]^t\Sigma^{-1}[\boldsymbol{x}-\boldsymbol{\mu}]\\ &=\begin{bmatrix}\boldsymbol{x}_1^t-\boldsymbol{\mu}_1^t & \boldsymbol{x}_2^t-\boldsymbol{\mu}_2^t\end{bmatrix}\begin{bmatrix}\Sigma_{11} & \Sigma_{12}\\ \Sigma_{21} & \Sigma_{22}\end{bmatrix}^{-1}\begin{bmatrix}\boldsymbol{x}_1-\boldsymbol{\mu}_1\\ \boldsymbol{x}_2-\boldsymbol{\mu}_2\end{bmatrix}=Q_1+Q_2\end{aligned} \tag{2.1.76}$$

여기서 Q_1과 Q_2는 각각 다음과 같다.

$$Q_1 \doteq [\boldsymbol{x}_1-\boldsymbol{\mu}_1]^t\Sigma_{11}^{-1}[\boldsymbol{x}_1-\boldsymbol{\mu}_1] \tag{2.1.77}$$

$$Q_2 \doteq \left\{[\boldsymbol{x}_2-\boldsymbol{\mu}_2]-\Sigma_{21}\Sigma_{11}^{-1}[\boldsymbol{x}_1-\boldsymbol{\mu}_1]\right\}^t\Sigma_{22\cdot1}^{-1}\left\{[\boldsymbol{x}_2-\boldsymbol{\mu}_2]-\Sigma_{21}\Sigma_{11}^{-1}[\boldsymbol{x}_1-\boldsymbol{\mu}_1]\right\} \tag{2.1.78}$$

또한, 식 (2.1.38)에서 알 수 있듯이, 다음 식이 성립한다.

$$|\,\Sigma\,| = |\,\Sigma_{11}\,|\cdot|\,\Sigma_{22\cdot1}\,| \tag{2.1.79}$$

식 (2.1.76)과 식 (2.1.79)에서 알 수 있듯이, 다음 식이 성립한다.

$$n(\boldsymbol{x} \mid \boldsymbol{\mu}, \Sigma) = n(\boldsymbol{x}_1 \mid \boldsymbol{\mu}_1, \Sigma_{11}) \cdot n(\boldsymbol{x}_2 \mid \boldsymbol{\mu}_2 + \Sigma_{21}\Sigma_{11}^{-1}[\boldsymbol{x}_1 - \boldsymbol{\mu}_1],\ \Sigma_{22\cdot 1}) \tag{2.1.80}$$

즉, 확률벡터 $\boldsymbol{x}_1$의 주변확률분포(marginal probability distribution)는 $\mathcal{N}((\boldsymbol{\mu}_1, \Sigma_{11})$이고 $\boldsymbol{x}_1$이 주어진 조건 하에서 확률벡터 $\boldsymbol{x}_2$의 조건부확률분포는 $\mathcal{N}\left(\boldsymbol{\mu}_2 + \Sigma_{21}\Sigma_{11}^{-1}[\boldsymbol{x}_1 - \boldsymbol{\mu}_1],\ \Sigma_{22\cdot 1}\right)$이다. 따라서, 다음 식들이 성립한다.

$$\boldsymbol{x}_1 \overset{d}{\sim} \mathcal{N}(\boldsymbol{\mu}_1, \Sigma_{11}) \tag{2.1.81}$$

$$\boldsymbol{x}_2 \mid \boldsymbol{x}_1 \overset{d}{\sim} \mathcal{N}(\boldsymbol{\mu}_2 + \Sigma_{21}\Sigma_{11}^{-1}[\boldsymbol{x}_1 - \boldsymbol{\mu}_1],\ \Sigma_{22\cdot 1}) \tag{2.1.82}$$

다변량정규확률분포 $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$를 따르는 확률벡터 $\boldsymbol{x}$와 행렬 A에 대해서 다음 식이 성립함을 쉽게 알 수 있다.

$$A\boldsymbol{x} \overset{d}{\sim} \mathcal{N}(A\boldsymbol{\mu}, A\Sigma A^t) \tag{2.1.83}$$

예제 2.1.7 다음 행렬을 살펴보자.

$$A = \begin{bmatrix} 1 & 0 \\ \rho & \sqrt{1-\rho^2} \end{bmatrix} \tag{1}$$

다음 식이 성립함은 명백하다.

$$AA^t = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} \tag{2}$$

서로 독립이며 표준정규확률분포를 따르는 확률변수들 z_1과 z_2으로 구성된 확률벡터를 $\boldsymbol{z} = [z_1, z_2]^t$로 표기하고, 확률벡터 $\boldsymbol{x} = [x_1, x_2]^t$를 다음과 같이 정의하자.

$$\boldsymbol{x} = A\boldsymbol{z} \tag{3}$$

다음 식들이 성립한다.

$$x_1 = z_1, \quad x_2 = \rho z_1 + \sqrt{1-\rho^2} z_2 \tag{4}$$

식 (2.1.83)에서 알 수 있듯이, 다음 식이 성립한다.

$$\boldsymbol{x} \overset{d}{\sim} \mathcal{N}\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1 & \rho\\ \rho & 1\end{bmatrix}\right) \tag{5}$$

다음 행렬을 살펴보자.

$$B = \begin{bmatrix}\sqrt{\rho} & \sqrt{1-\rho} & 0\\ \sqrt{\rho} & 0 & \sqrt{1-\rho}\end{bmatrix} \tag{6}$$

다음 식이 성립함은 명백하다.

$$BB^t = \begin{bmatrix}1 & \rho\\ \rho & 1\end{bmatrix} \tag{7}$$

서로 독립이며 표준정규확률분포를 따르는 확률변수들 z_1, z_2와 z_3로 구성된 확률벡터를 $\boldsymbol{z} = [z_1, z_2, z_3]^t$로 표기하고, 확률벡터 $\boldsymbol{y} = [y_1, y_2]^t$를 다음과 같이 정의하자.

$$\boldsymbol{y} = B\boldsymbol{z} \tag{8}$$

다음 식들이 성립한다.

$$y_1 = \sqrt{\rho}z_1 + \sqrt{1-\rho}z_2, \quad y_2 = \sqrt{\rho}z_1 + \sqrt{1-\rho}z_3 \tag{9}$$

식 (2.1.83)에서 알 수 있듯이, 다음 식이 성립한다.

$$\boldsymbol{y} \overset{d}{\sim} \mathcal{N}\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1 & \rho\\ \rho & 1\end{bmatrix}\right) \tag{10}$$

■

예제 2.1.8 다음 행렬을 살펴보자.

$$A = \begin{bmatrix}\sqrt{\rho} & \sqrt{1-\rho} & 0 & 0 & \cdots & 0\\ \sqrt{\rho} & 0 & \sqrt{1-\rho} & 0 & \cdots & 0\\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots\\ \sqrt{\rho} & 0 & 0 & 0 & \cdots & \sqrt{1-\rho}\end{bmatrix} \tag{1}$$

다음 식이 성립한다.

$$AA^t = \begin{bmatrix} 1 & \rho & \cdots & \rho \\ \rho & 1 & \cdots & \rho \\ \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \cdots & 1 \end{bmatrix} \tag{2}$$

식 (2) 의 상관구조가 성립하면, 인트라클래스상관계수 (intraclass correlation)를 갖는다고 한다.

서로 독립인 표준정규확률변수들 $z_0, z_1, \cdots, z_p$ 에 대해서, 다음 확률변수들을 정의하자.

$$x_j \doteq \sqrt{\rho} z_0 + \sqrt{1-\rho} z_j, \quad (j = 1, 2, \cdots, p) \tag{3}$$

식 (2.1.83) 에서 알 수 있듯이, 확률벡터 $\boldsymbol{x} = [x_1, x_2, \cdots, x_p]^t$ 는 인트라상관계수를 갖는다. ■

제2.2절 난수벡터와 난수행렬

2.2.1 종속원소발생기

확률벡터 $\boldsymbol{x} = [x_1, x_2, \cdots, x_p]^t$ 의 결합확률밀도함수 $f(\boldsymbol{x})$ 는 다음 식을 만족한다.

$$f(\boldsymbol{x}) = f(x_1) \prod_{i=2}^{p} f(x_i \ | x_1, \cdots, x_{i-1}) \tag{2.2.1}$$

여기서 $f(x_1)$ 은 확률변수 x_1 의 주변확률밀도함수 (marginal probability density function)이고 $f(x_i \mid x_1, \cdots, x_{i-1})$ 는 $x_1, x_2, \cdots, x_{i-1}$ 가 주어진 조건 하에서 확률변수 x_i 의 조건부확률밀도함수이다. 따라서, 조건부확률분포를 아는 경우에는 식 (2.2.1) 을 사용해서 난수벡터 (random vector)를 생성할 수 있다. 즉, 다음과 같이 종속원소발생기 (dependent components generator)를 사용해서 난수벡터를 생성할 수 있다.

알고리즘 2.2.1: 종속원소발생기

(1단계) 확률밀도함수가 $f(x_1)$ 인 난수 x_1 을 발생한다.

(2단계) 각 $i(= 2, 3, \cdots, p-1)$ 에 대해서 $x_1, x_2, \cdots, x_{i-1}$ 가 주어진 조건 하에서 확률변수 x_i 의 조건부확률분포로부터 난수 x_i 를 발생한다.

(3단계) 제1단계와 제2단계의 결과를 결합해서 $\boldsymbol{x} = [x_1, x_2, \cdots, x_p]^t$를 난수벡터로 한다.

예를 들어, 다변량정규확률분포나 Markov체인과 같은 경우에는 알고리즘 2.2.1을 사용해서 난수벡터를 생성할 수도 있다. 그러나, 다변량인 경우에는 채택기각법을 사용하는 것이 편리한 경우가 많다. 또한, 차수 p가 큰 경우에는 정확한 난수벡터를 생성하는 것이 어려운 경우가 많으므로, MCMC와 같은 점근적 방법을 사용한다.

예제 2.2.1 종속원소발생기를 사용해서 정규난수벡터들을 생성하기 위해서 다음 MATLAB 프로그램 DependentComponentsGenerator101.m을 실행해 보자.

```
% ---------------------------------------------------------------------
% Filename: DependentComponentsGenerator101.m
% Generating Random Vector by Dependent Components Generator
% Programmed by CBS
% ---------------------------------------------------------------------
clear, clf, close all
N = 200;
x = zeros(2,N);
mu = [ 0.4 ; -0.8 ]
Sigma = [ 1 -1.28 ; -1.28 4 ]
beta = Sigma(2,1)/Sigma(1,1)
sigma11 = sqrt(Sigma(1,1))
sigma22 = sqrt(Sigma(2,2))
Sigma221 = Sigma(2,2) - Sigma(2,1)/Sigma(1,1)*Sigma(1,2)
sigma221 = sqrt(Sigma221)
rand('twister',5489)
for jj=1:N
    x(1,jj) = random('Normal',mu(1),sigma11,1,1);
    mu2 = mu(2) + beta*(x(1,jj)-mu(1));
    x(2,jj) = random('Normal',mu2,sigma221,1,1);
end
% Mean and Covariance Matrix
xmean = mean(x')
xcov = cov(x')
% Contour of the True PDF
R = max(diag(Sigma))*4;
[xgd, ygd] = meshgrid(mu(1)-R:0.05:mu(1)+R,mu(2)-R:0.05:mu(2)+R);
iSig = Sigma^(-1)
dgd = iSig(1,1)*(xgd-mu(1)).^2 + iSig(2,2)*(ygd-mu(2)).^2 ...
      + 2*iSig(1,2)*(xgd-mu(1)).*(ygd-mu(2));
f = 1/(2*pi*sqrt(det(Sigma)))*exp(-1/2*dgd);
contour(xgd,ygd,f,'linewidth',2)
colormap(bone)
colorbar('location','westoutside')
set(gca,'fontsize',11,'fontweigh','bold')
hold on
% Scatter Plot
xx1 = (-10:0.05:10)';
xx2 = mu(2) + beta*(xx1-mu(1));
plot(x(1,:),x(2,:),'k.',xx1,xx2,'r-','linewidth',2)
plot([mu(1) mu(1)],[-10 10 ],'b-',[-10 10],[mu(2) mu(2)],'b-')
axis([ -4 4 -8 6 ])
hold off
```

```
saveas(gcf,'DependentComponentsGenerator101','epsc')
save('DependentComponentsGenerator101','x')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 다음 2변량 정규확률분포에서 난수벡터들을 200개를 생성한다.

$$\boldsymbol{x} \overset{d}{\sim} \mathcal{N}\left(\begin{bmatrix} 0.4 \\ 0.8 \end{bmatrix}, \begin{bmatrix} 1 & -1.28 \\ -1.28 & 4 \end{bmatrix} \right) \tag{1}$$

식 (2.1.81)과 식 (2.1.82)에서 알 수 있듯이, 확률변수 x_1과 조건부확률변수 $x_2 \,|\, x_1$의 확률분포는 각각 다음과 같다.

$$x_1 \overset{d}{\sim} \mathcal{N}\left(0.4, 1^2\right) \tag{2}$$

$$x_2 \,|\, x_1 \overset{d}{\sim} \mathcal{N}\left(-0.8 - 1.28[x_1 - 0.4], 1.5367^2\right) \tag{3}$$

이렇게 발생된 난수벡터들의 표본평균벡터와 표본분산공분산행렬은 각각 다음과 같다.

$$\hat{\boldsymbol{\mu}} = \begin{bmatrix} 0.4359 \\ -0.9193 \end{bmatrix}, \quad \hat{\Sigma} = \begin{bmatrix} 1.0845 & -1.4067 \\ -1.4067 & 4.0020 \end{bmatrix} \tag{4}$$

이 MATLAB프로그램을 실행하면, 식 (2)와 식 (3), 그리고 알고리즘 2.2.1에 의해서 생성된 2변량 정규난수벡터들의 산포도인 그림 2.2.1이 그려진다. 그림 2.2.1에서 각 점은 정규난수벡터를 나타내고, 청색 가는 실선들은 직선 $x_1 = 0.4$와 직선 $x_2 = -0.8$을 나타내고 적색 실선은 조건부기대값, 즉 다음 식을 나타낸다.

$$x_2 = -0.8 - 1.28[x_1 - 0.4] \tag{5}$$

또한, 타원은 이 2변량 정규확률분포의 레벨곡선이고, 좌측의 회색막대는 이 레벨곡선의 수준을 나타낸다. ■

2.2.2 일양난수벡터

다음과 같이 단위초구(unit hyperball) $B_p = \{\boldsymbol{x} \in R^p \mid \|\boldsymbol{x}\| \leq 1\}$ 에서 일양난수벡터를 발생시킬 수 있다.

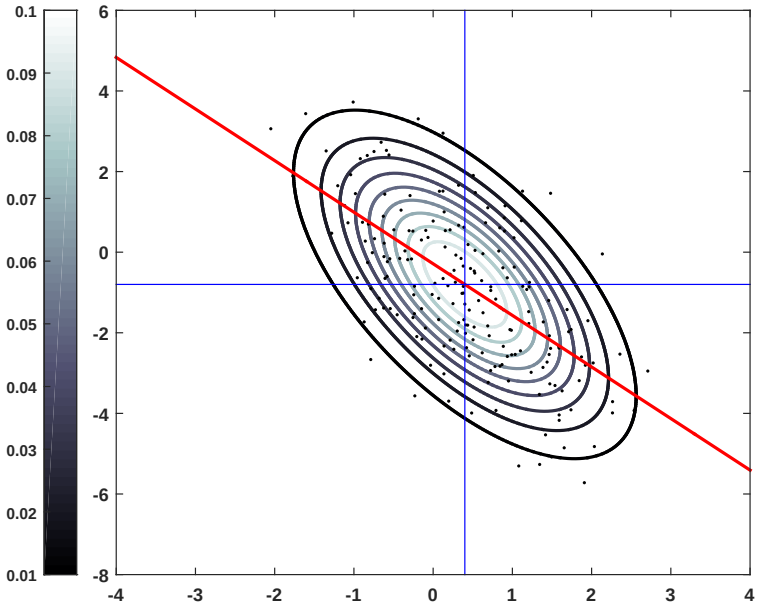


그림 2.2.1. 종속원소발생기와 정규난수벡터

알고리즘 2.2.2: 일양난수벡터를 생성하는 채택기각법

(1단계) 지지대가 $(0,1)$인 일양난수들 $u_1, u_2, \cdots, u_p$에 대해서 다음 값들을 계산한다.

$$x_1 = 1 - 2u_1, \cdots, x_p = 1 - 2u_p, \quad R = \sum_{i=1}^{p} x_i^2$$

(2단계) 만약 $R \leq 1$이면, $\boldsymbol{x} = [x_1, x_2, \cdots, x_p]^t$를 원하는 일양난수벡터로 채택한다.
그렇지 않으면 제1단계로 돌아간다.

알고리즘 2.2.2는 효율적이 아니다. 이 채택기각법의 채택율은 단위초구의 부피를 2^p으로 나눈 것이다. 차원이 p인 단위초구의 부피는 $\pi^{p/2}/\Gamma([p+1]/2)$이다. 즉, 채택율은 다음과 같다.

$$\frac{1}{C} = \frac{\pi^{p/2}}{p2^{p-1}\Gamma(p/2)} \tag{2.2.2}$$

예제 2.2.2 식 ??를 그림으로 나타내기 위해서, 다음 MATLAB프로그램 EfficiencyAcceptRejectUnitBall101.m을 실행하라.

```
% ------------------------------------------------------------------------
% Filename: EfficiencyAcceptRejectUnitBall101.m
```

```
% Efficiency of Accept-Reject Method of Unit Hyperball
% Programmed by CBS
% ------------------------------------------------------------------------
clear all, close all
pp = 1:15;
Efficiency = pi.^(pp/2)./( pp.*2.^(pp-1).*gamma(pp/2) );
% Ploting
plot(pp,Efficiency,'r--*','linewidth',1.5)
set(gca,'fontsize',11,'fontweigh','bold','xtick',0:1:15)
xlabel('\bf p'), ylabel('\bf Efficiency')
saveas(gcf,'EfficiencyAcceptRejectUnitBall101','epsc')
save('EfficiencyAcceptRejectUnitBall101','Efficiency')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 그림 2.2.2 를 출력한다. 그림 2.2.2 에서 알 수 있듯이, 차원 p가 커짐에 따라 급속히 감소한다. 특히 p가 6이상이면 효율성은 10% 미만이다.

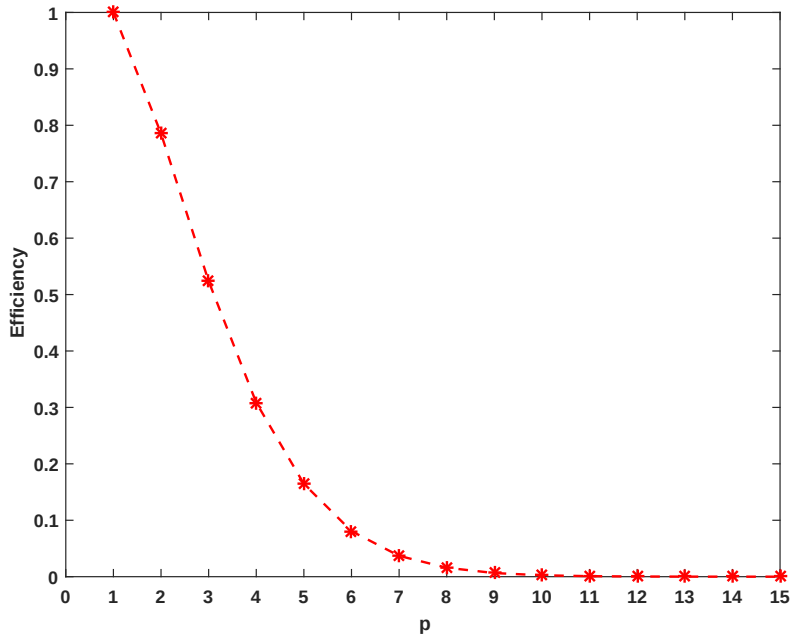


그림 2.2.2. 일양난수벡터의 채택율

다음 명제들은 효율적으로 단위초구 B_p에서 일양난수벡터를 발생시키기 위해서 Marsaglia [30]가 제시한 것이다.

명제 2.2.1

서로 독립이고 지지대가 $(-1,1)$인 일양확률변수 $v_1, v_2, \cdots, v_p$로 구성된 확률벡터 $\boldsymbol{v} = [v_1, v_2, \cdots, v_p]^t$가 조건 $\|\boldsymbol{v}\| < 1$을 만족하면, 확률벡터 $\boldsymbol{w} = \boldsymbol{v}/\|\boldsymbol{v}\|$는 초구면 (hypersphere) $S_p = \{\boldsymbol{w} \in R^p \mid \|\boldsymbol{w}\| = 1\}$ 에서 일양분포한다.

명제 2.2.2

서로 독립인 표준정규확률변수 $x_1, x_2, \cdots, x_p$로 구성된 확률벡터 $\boldsymbol{x} = [x_1, \cdots, x_p]^t$에 대해서 확률벡터 $\boldsymbol{y} = \boldsymbol{x}/\|\boldsymbol{x}\|$는 초구면 S_p에서 일양분포한다.

단위초구 상에서 일양분포하는 점의 반경을 R이라고 하면, 확률변수 R의 확률분포함수는 다음과 같다.

$$F_R(r) = Pr(R \le r) = \frac{\pi^{p/2}/\Gamma([p+1]/2)r^p}{\pi^{p/2}/\Gamma([p+1]/2)} = r^p \tag{2.2.3}$$

따라서, 다음과 같이 단위초구 B_p에서 일양난수벡터를 발생시킬 수 있다.

알고리즘 2.2.3: 단위초구 상에서 일양난수벡터를 생성하는 방법

(1단계) 정규난수들 $x_1, x_2, \cdots, x_p$을 생성한다.

(2단계) 지지대가 $(0, 1)$인 일양난수 u를 생성하고, $r = u^{1/p}$를 계산한다.

(3단계) 벡터 $\boldsymbol{x} = [x_1, x_2, \cdots, x_p]^t$에 대해서 벡터 $\boldsymbol{y} = r\boldsymbol{x}/\|\boldsymbol{x}\|$를 원하는 일양난수벡터로 채택한다.

예제 2.2.3 알고리즘 2.2.3를 사용해서 단위초구 상에서 일양난수벡터들을 생성하기 위해서 다음 MATLAB프로그램 MarsagliaUniformHypersphere101.m을 실행해 보자.

```
% ------------------------------------------------------------------
% Filename: MarsagliaUniformHypersphere101.m
% Generating Multivariate Uniform Didt on Unit Hypersphere
% by Marsaglia Method (1972)
% Programmed by CBS
% ------------------------------------------------------------------
clear, clf, close all
N = 10000
p = 3;
rng(5489,'twister')
x = randn(p,N);
r = (rand(1,N)).^(1/p);
for jj = 1:N
    normx(jj) = norm(x(:,jj),2);
end
y = (ones(p,1)*r).*x./(ones(p,1)*normx);
% Plotting
[a,b,c] = sphere;
```

```
surf(a,b,c)
daspect([1 1 1])
hold on
plot3(y(1,:),y(2,:),y(3,:),'k.','linewidth',3)
set(gca,'fontsize',11,'fontweigh','bold')
axis equal
hold off
saveas(gcf,'MarsagliaUniformHypersphere101','epsc')
save('MarsagliaUniformHypersphere101','y')
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 단위구면 상에서 일양난수벡터들을 10000 개를 생성한다. 또한, 이 일양난수벡터들의 산포도인 그림 2.2.3이 그려진다. 이 그림에서 단위구를 그리기 위해서는 MATLAB함수 sphere.m를 사용하였다. 그림 2.2.3에서 확인할 수 있듯이, 이 일양난수벡터들은 단

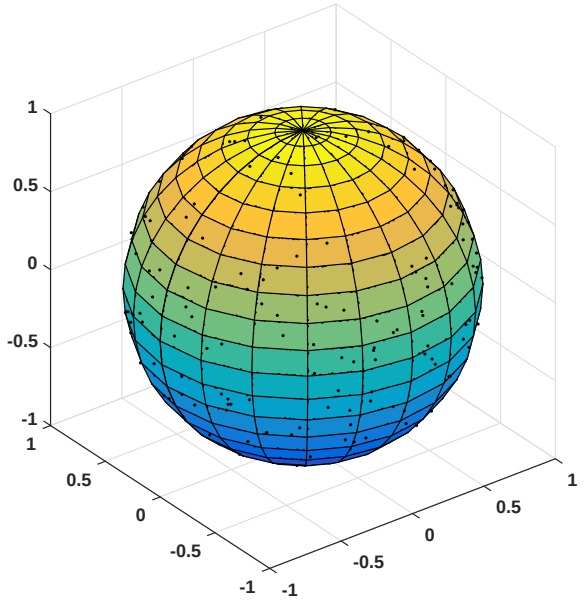


그림 2.2.3. 단위구면 상의 정규난수벡터

2.2.3 정규난수벡터

예제 2.2.4 MATLAB을 이용해서 2변량 정규확률벡터의 결합확률밀도함수, 등고선도(contour map), 결합누적확률분포함수, 그리고 정규난수벡터를 살펴보기 위해서 다음 MATLAB 프로그램 MultivariateNormalRV101.m을 실행해 보자.

```
% ------------------------------------------------------------------
% Filename: MultivariateNormal201.m
% Multivariate Normal Random Vector
```

```
% Programmed by CBS
% ------------------------------------------------------------------
clear, clf, close all
rng(5489,'twister')
mu = [ 0.4  -0.8 ]
Sigma = [ 1 -1.28 ; -1.28 4 ]
x1 = -3.5:0.2:3.5; x2 = -7:0.2:7;
[X1,X2] = meshgrid(x1,x2);
% PDF
F = mvnpdf([X1(:) X2(:)],mu,Sigma);
F = reshape(F,length(x2),length(x1));
subplot(2,2,1)
surf(x1,x2,F);
set(gca,'fontsize',11,'fontweigh','bold')
colormap(gray)
caxis([min(F(:))-.5*range(F(:)),max(F(:))]);
axis([-3.3 3.5 -7 7  0 0.1 ])
xlabel('\bf x1'); ylabel('\bf x2'); zlabel('\bf PDF');
% Conour
Fmax = max(max(F))
subplot(2,2,2)
contour(x1,x2,F,Fmax*(0.1:0.1:1));
set(gca,'fontsize',11,'fontweigh','bold')
axis([-3.3 3.5 -7 7 ])
xlabel('\bf x1'); ylabel('\bf x2');
% CDF
F = mvncdf([X1(:) X2(:)],mu,Sigma);
F = reshape(F,length(x2),length(x1));
subplot(2,2,3)
surf(x1,x2,F);
set(gca,'fontsize',11,'fontweigh','bold')
colormap(gray)
caxis([min(F(:))-.5*range(F(:)),max(F(:))]);
axis([-3.3 3.5 -7 7  0 1 ])
xlabel('\bf x1'); ylabel('\bf x2'); zlabel('\bf CDF');
% Random Vectors
Xrnd = mvnrnd(mu,Sigma,100);
subplot(2,2,4)
plot(Xrnd(:,1),Xrnd(:,2),'k.','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
axis([-3.3 3.5 -7 7 ])
xlabel('\bf x1'); ylabel('\bf x2');
saveas(gcf,'MultivariateNormalRV101','epsc')
save('MultivariateNormalRV101','Xrnd')
% End of program
% ------------------------------------------------------------------
```

다변량정규확률분포에 관한 MATLAB함수들로는 mvnpdf, mvncdf, mvnrnd 등이 있다.

이 MATLAB프로그램을 실행하면, 다음 2변량 정규확률분포의 결합확률밀도함수, 이 결합확률밀도함수의 등고선도, 그리고 결합누적확률분포함수를 그린다.

$$\boldsymbol{x} \overset{d}{\sim} \mathcal{N}\left(\begin{bmatrix} 0.4 \\ 0.8 \end{bmatrix}, \begin{bmatrix} 1 & -1.28 \\ -1.28 & 4 \end{bmatrix} \right) \tag{1}$$

또한, 이 확률분포에서 100개 정규난수벡터들을 생성하고, 이들의 등고선도를 그린다. 이

그래프들이 그림 2.2.4에 그려져 있다. ■

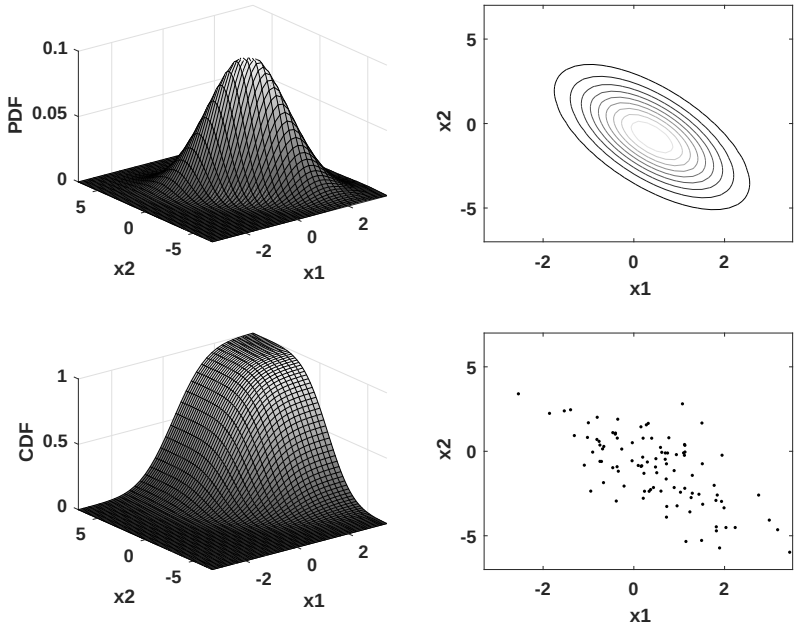


그림 2.2.4. 정규확률벡터와 MATLAB

예제 2.2.5 R을 이용해서 2변량 정규확률벡터로부터 난수벡터들을 생성하고 이들의 산점도와 히스토그램들을 그리기 위해서 다음 R프로그램 MultivariateNormalRV101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------------
# Filename: MultivariateNormalRV101R.R
# Multivariate Normal Random Vector
# Programmed by CBS
# ------------------------------------------------------------------------------
# install.packages("mvtnorm")
library(mvtnorm)
( mu <- matrix(c(0.4,0.8),2,1) )
( Sigma <- matrix(c(1,-1.28,-1.28,4),2,2) )

# Generating Normal random vectors
nSim <- 100
Xrand <- rmvnorm(nSim, mu, Sigma)

# Generate Density Function and its Contour
x1 <- seq(-2.5,3.0,length.out=21)
x2 <- seq(-5.0,5.5,length.out=21)
zN <- matrix(0,length(x1)*length(x2),ncol=1)
xN <- zN
yN <- zN
for (ii in 1:length(x1)) {
    for (jj in 1:length(x2)) {
        dum <- (ii-1)*length(x2)+jj
        xN[dum] <- x1[ii]
        yN[dum] <- x2[jj]
        xdum <- cbind(x1[ii],x2[jj])
```

```
        zN[dum] <- dmvnorm(xdum,mu,Sigma)
    }
}

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('MultivariateNormalRV101R.eps')  # Start to save figure
RVdata1 <- data.frame(Xrand)
RVdata2 <- data.frame(xN,yN,zN)
plot11 <- ggplot(RVdata1, aes(x=Xrand[,1],y=Xrand[,2])) +
          geom_point(col="red",lwd=1.2) +
          xlab(expression(x_1)) + ylab(expression(x_2)) +
          geom_density2d()
plot12 <- ggplot(RVdata1, aes(x=Xrand[,1])) +
          geom_histogram(bins=20,fill="white",color="black")+
          xlab(expression(x_1))
plot13 <- ggplot(RVdata1, aes(x=Xrand[,2])) +
          geom_histogram(bins=20,fill="yellow",color="dark blue")+
          xlab(expression(x_2))
plot14 <- ggplot(RVdata2,aes(x=xN,y=yN,fill=zN)) +
          geom_tile() +
          geom_raster()
pushViewport(viewport(layout=grid.layout(2,2)))
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot13, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot14, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()             # End to save figure
# ------------------------------------------------------------------------------
```

R패키지 mvtnorm을 사용해서, 다음 정규확률분포로부터 정규난수벡터를 생성하자.

$$\boldsymbol{x} \overset{d}{\sim} \mathcal{N}\left(\begin{bmatrix} 0.4 \\ 0.8 \end{bmatrix}, \begin{bmatrix} 1 & -1.28 \\ -1.28 & 4 \end{bmatrix} \right) \tag{1}$$

이 R프로그램을 실행하면, 이 2변량 정규확률분포에서 100개 정규난수벡터들을 생성하고, 이들의 산점도와 각 성분 확률변수의 히스토그램을 그린다. 이 그래프들이 그림 2.2.5에 그려져 있다. ■

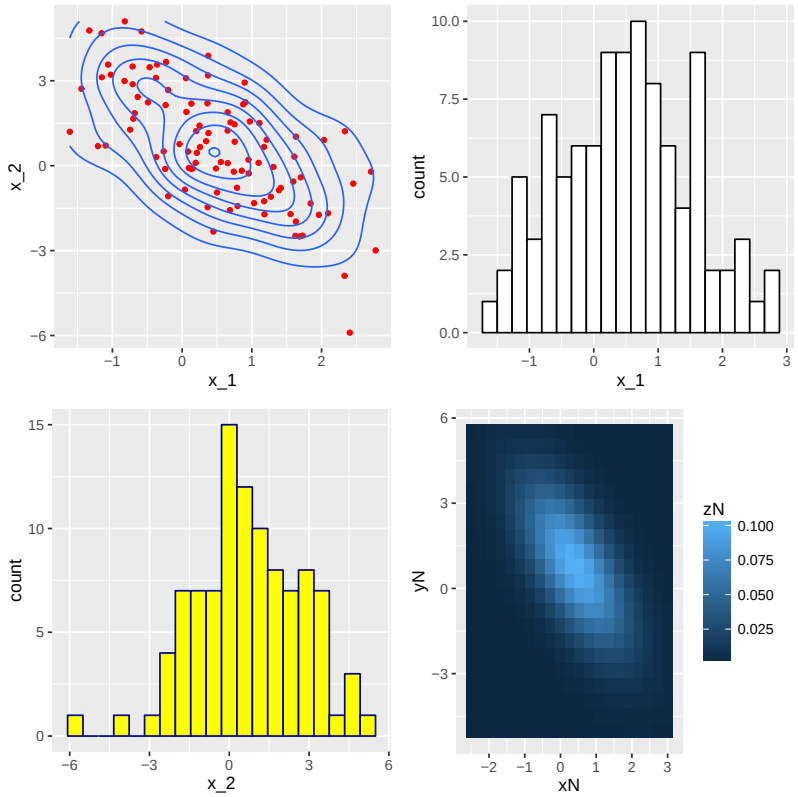


그림 2.2.5. 정규확률벡터와 R

2.2.4 Dirichlet난수벡터

모수벡터가 $\boldsymbol{a} = [a_1, a_2, \cdots, a_{p+1}]^t$ 인 Dirichlet확률분포 $Dirichlet(\boldsymbol{a})$ 의 확률밀도함수는 다음과 같다.

$$f(\boldsymbol{x}) = \frac{\Gamma\left(\sum_{i=1}^{p+1} a_i\right)}{\prod_{i=1}^{p+1} \Gamma(a_i)} \left[\prod_{i=1}^{p} x_i^{a_i - 1}\right] \left[1 - \sum_{i=1}^{p} x_i\right]^{a_{p+1}-1} \tag{2.2.4}$$

여기서 $\boldsymbol{x} = [x_1, x_2, \cdots, x_{p+1}]^t$ 는 다음 식들을 만족한다.

$$x_k \geq 0, \quad (k = 1, 2, \cdots, p), \qquad \sum_{i=1}^{p} x_i \leq 1 \tag{2.2.5}$$

각 k에 대해서 다음 식들이 성립한다.

$$E(x_k) = \frac{a_k}{a_0}, \quad Var(x_k) = \frac{a_k[a_0 - a_k]}{a_0^2[a_0 + 1]}, \quad Cov(x_i, x_j) \frac{-\alpha_i \alpha_j}{\alpha_0^2(\alpha_0 + 1)}, \quad (i \neq j) \tag{2.2.6}$$

여기서 $a_0 = \sum_{i=1}^{p+1} a_i$ 이다. 즉, Dirichlet확률분포는 베타분포를 확장한 것이다.

확률변수들 $y_1, y_2, \cdots, y_{p+1}$ 이 서로 독립이고, 확률변수 y_k 가 감마확률분포 $Gamma(a_k, 1)$ 를 따른다고 하고, 다음 확률변수들을 정의하자.

$$x_k \doteq \frac{y_k}{\sum_{i=1}^{p+1} y_i}, \quad (k = 1, 2, \cdots, p+1) \tag{2.2.7}$$

예제 1.5.23의 방법을 사용해서 다음 식을 유도할 수 있다.

$$\boldsymbol{x} = [x_1, x_2, \cdots, x_{p+1}]^t \overset{d}{\sim} Dirichlet(\boldsymbol{a}) \tag{2.2.8}$$

예제 2.2.6 감마난수들로부터 Dirichlet난수벡터들을 생성하기 위해서 다음 MATLAB프로그램 DirichletRandomVector101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: DirichletRandomVector101.m
%  Dirichlet Random Vectors from Gamma Random Numbers
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all
p = 2, p1 = p+1                          % Dimension
a = [ 3 2 4 ]                            % Parameter Vetor
N = 500                                  % No of Random Numbers
rng(5489,'twister')
for ii=1:p+1
     y(ii,:) = randg(a(ii),1,N);
end
xran = y./(ones(p+1,1)*sum(y,1));          % Dirichlet Random Vector
% (another method) xran = y./repmat(sum(y,1),p+1,1);
% Plotting
x1 = linspace(0,1,101);
x2 = linspace(0,1,101);
[X1,X2] = ndgrid(x1,x2);
X3 = 1 - X1 - X2;
bad = (X1+X2 > 1); X1(bad) = NaN; X2(bad) = NaN; X3(bad) = NaN;
Const = exp(sum(gammaln(a))-gammaln(sum(a)));
F = (X1.^(a(1)-1) .* X2.^(a(2)-1) .* X3.^(a(3)-1))/Const;
surf(X1,X2,X3,F);
set(gca,'fontsize',11,'fontweigh','bold')
colormap([1 1 0])
xlabel('x1'); ylabel('x2'); zlabel('x3');
hold on
plot3(xran(1,:),xran(2,:),xran(3,:),'k.','linewidth',3)
az = 134;
el = 37;
view(az, el);
hold off
saveas(gcf,'DirichletRandomVector101','epsc')
save('DirichletRandomVector101','F')
% End of program
```

```
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, Dirichlet확률분포 $Dirichlet(3,2,4)$에서 난수벡터들 200개를 발생한다. 또한 그림 2.2.6가 출력된다. 그림 2.2.6에서 황색 평면은 식 $x_1+x_2+x_3=1$을 나타내고, 이 황색 평면 위의 흑색 점들은 생생된 Dirichlet난수벡터를 나타낸다. ■

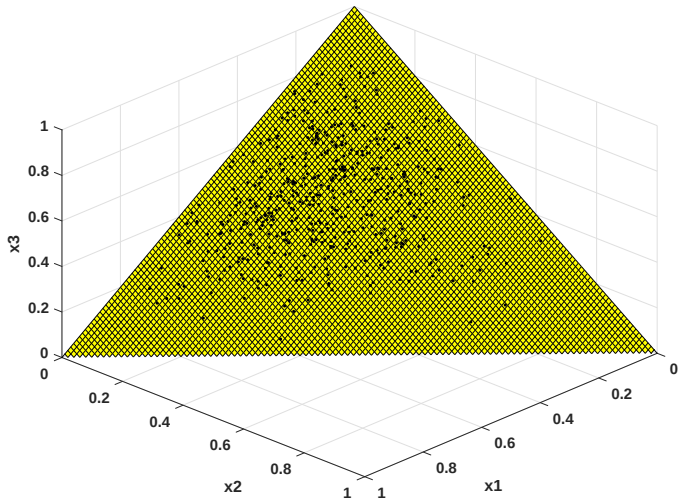


그림 2.2.6. Dirichlet난수벡터

예제 2.2.7 Dirichlet난수벡터들을 생성하기 위해서 다음 R프로그램 DirichletRandomVector101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: DirichletRandomVector101R.R
# Dirichlet random Vector
# Programmed by CBS
# ------------------------------------------------------------------------
# install.packages("MCMCpack")
library(MCMCpack)
set.seed(11)
nSim <- 500
a <- c(3,2,4)
X <- rdirichlet(nSim,a)

# Plotting
# install.packages("scatterplot3d")
library(scatterplot3d)
setEPS()
plot.new()
postscript('DirichletRandomVector101R.eps')  # Start to save figure
Xdata <- data.frame(X)
attach(Xdata)
S3D <- scatterplot3d(X[,1],X[,2],X[,3],
                     pch=20,
                     xlab="x1",ylab="x2",zlab="x3",
                     type="p")
FitPlane <- lm(X[,3] ~ X[,1]+X[,2])
```

```
S3D$plane3d(FitPlane)
dev.off()             # End to save figure
# ------------------------------------------------------------------------
```

이 R프로그램을 실행하면, Dirichlet확률분포 $Dirichlet(3,2,4)$에서 난수벡터 500개를 생성한다. 그림 2.2.7에서 이 난수벡터들 흑색 점들로 표기된다. 이 점들이 평면 $x_1+x_2+x_3=1$ 위에 있음을 확인할 수 있다. ■

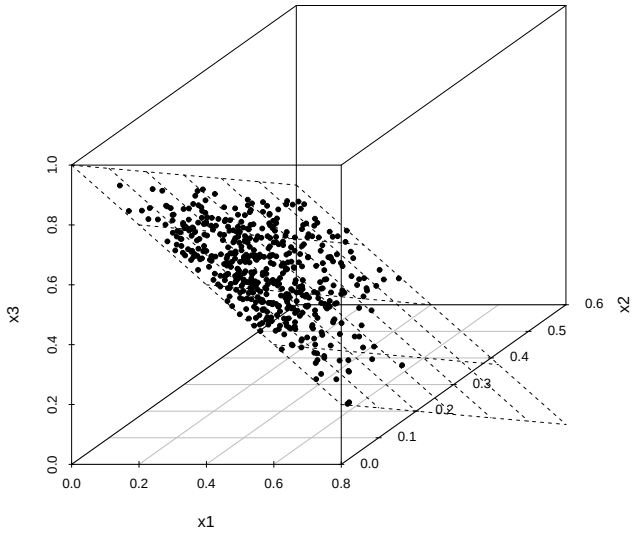


그림 2.2.7. Dirichlet난수벡터

예제 2.2.8 Dirichlet확률분포의 등고선도를 그리기 위해서 다음 R프로그램 DirichletDensity101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: DirichletDensity101R.R
# Dirichlet Probability Density
# Programmed by CBS
# ------------------------------------------------------------------------
# install.packages("MCMCpack")
library(MCMCpack)
a <- c(3,2,4)

# Generate Density Function and its Contour
x1 = seq(0,1,length.out=100)
x2 = seq(0,1,length.out=100)
z = matrix(0,length(x1),length(x2))
for (ii in 1:length(x1)) {
    for (jj in 1:length(x2)) {
        if(x1[ii]+x2[jj] > 1){
           z[ii,jj] <- 0
```

```
        }  else {
                xx = cbind(x1[ii],x2[jj],1-x1[ii]-x2[jj])
                z[ii,jj] = ddirichlet(xx,a)
        }
    }
}

# Plotting
setEPS()
plot.new()
postscript('DirichletDensity101R.eps')  # Start to save figure
image(x1,x2,z)
dev.off()            # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, Dirichlet확률분포 $Dirichlet(3,2,4)$의 등고선도가 그려진다. 이 등고선도가 그림 2.2.8에 수록되어 있다. ■

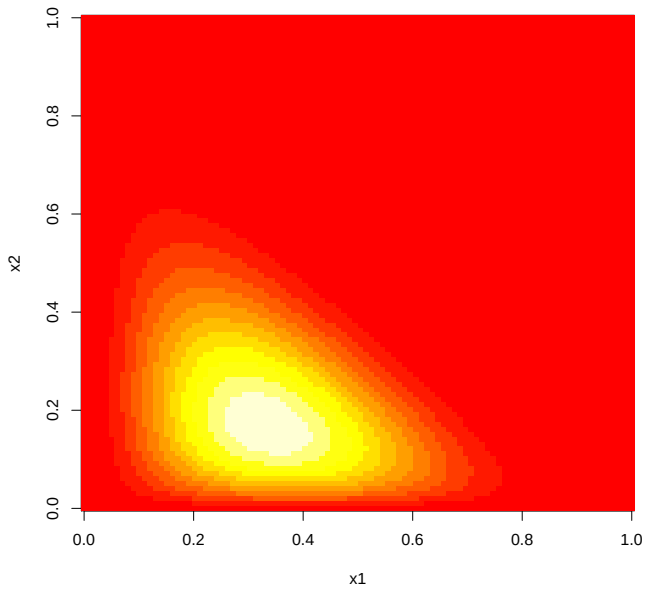


그림 2.2.8. Dirichlet난수벡터

2.2.5 다변량 t확률분포

확률벡터 $\boldsymbol{x} = [x_1, x_2, \cdots, x_p]^t$가 다음과 같은 결합확률밀도함수를 갖는다고 하자.

$$f(\boldsymbol{x}) = \frac{\Gamma(\frac{\nu+p}{2})}{[\pi\nu]^{p/2}\Gamma(\frac{\nu}{2})}\left[1+\frac{1}{\nu}\boldsymbol{x}^t\boldsymbol{x}\right]^{-[\nu+p]/2} \tag{2.2.9}$$

여기서 ν를 자유도 또는 형태모수(shape parameter)라 부르고, 이에 해당하는 확률분포를 $\boldsymbol{t}_\nu$로 표기한다. 평균벡터가 $\mathbf{0}$이고 분산공분산행렬이 I인 p변량 정규확률벡터 $\boldsymbol{z}$와 이와 독립인 $Gamma(\nu/2, 1/2)$ 확률분포, 즉 χ^2_ν 확률분포를 따르는 확률변수 S에 대해서 다음 식이 성립한다.

$$\sqrt{\frac{\nu}{S}}\boldsymbol{z} \overset{d}{\sim} \boldsymbol{t}_\nu \tag{2.2.10}$$

확률벡터 $\boldsymbol{y} = [y_1, y_2, \cdots, y_q]^t$의 확률분포가 q차원 $\boldsymbol{t}_\nu$이고, A는 $q \times p$ 행렬, 그리고 $\boldsymbol{\mu}$는 q차원 열벡터라고 하자. 벡터 $\boldsymbol{x} = \boldsymbol{\mu} + A\boldsymbol{y}$를 평균벡터가 $\boldsymbol{\mu}$이고 크기행렬(scale matrix)이 $\Sigma = AA^t$인 다변량 t확률분포라 부르고 $t_\nu(\boldsymbol{\mu}, \Sigma)$로 표기한다.

알고리즘 2.2.4: 다변량확률분포 $\boldsymbol{t}_\nu(\boldsymbol{\mu}, \Sigma)$에서 난수벡터 발생법

(1단계) 행렬 Σ의 Cholesky인자 A를 구한다. 즉, 식 $\Sigma = AA^t$을 만족하는 A를 구한다.

(2단계) 다변량정규확률분포 $\mathcal{N}_p(\mathbf{0}, I)$에서 정규난수벡터 $\boldsymbol{z}$을 생성하고, 감마확률분포 $Gamma(v/2, v/2)$로부터 감마난수 S를 발생시켜, 벡터 $\boldsymbol{y} = \sqrt{v/S}\boldsymbol{z}$를 계산한다.

(3단계) 벡터 $\boldsymbol{x} = \boldsymbol{\mu} + A\boldsymbol{y}$를 계산하면, $\boldsymbol{x}$가 $t_\nu(\boldsymbol{\mu}, \Sigma)$에서 생성된 난수벡터이다.

예제 2.2.9 MATLAB을 이용해서 2변량 t확률분포의 확률질량함수, 등고선도, 확률분포함수, 그리고 t난수벡터를 살펴보기 위해서 다음 MATLAB프로그램 MultiVariateTrv10101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
% Filename: MultiVariateT101.m
% Multivariate T Random Vector
% Programmed by CBS
% ------------------------------------------------------------------------
clear, clf, close all
rng(5489,'twister')
Rho = [1 .4 ; .4 1];                          % Correlation Matrix
df = 5;                                       % Degrees of Freedom
x1 = -3.5:0.2:3.5; x2 = -7:0.2:7;
[X1,X2] = meshgrid(x1,x2);
% PDF
F = mvtpdf([X1(:) X2(:)],Rho,df);
F = reshape(F,length(x2),length(x1));
subplot(2,2,1)
surf(x1,x2,F);
set(gca,'fontsize',11,'fontweigh','bold')
colormap('hsv')
```

```
caxis([min(F(:))-.5*range(F(:)),max(F(:))]);
axis([-3.3 3.5 -7 7  0 0.15 ])
xlabel('\bf x1'); ylabel('\bf x2'); zlabel('\bf PDF');
% Conour
Fmax = max(max(F))
subplot(2,2,2)
contour(x1,x2,F,Fmax*(0.1:0.1:1));
set(gca,'fontsize',11,'fontweigh','bold')
axis([-3.3 3.5 -7 7 ])
xlabel('\bf x1'); ylabel('\bf x2');
% CDF
F = mvtcdf([X1(:) X2(:)],Rho,df);
F = reshape(F,length(x2),length(x1));
subplot(2,2,3)
surf(x1,x2,F);
set(gca,'fontsize',11,'fontweigh','bold')
caxis([min(F(:))-.5*range(F(:)),max(F(:))]);
axis([-3.3 3.5 -7 7  0 1 ])
xlabel('\bf x1'); ylabel('\bf x2'); zlabel('\bf CDF');
% Random Vectors
Xrnd = mvtrnd(Rho,df,100);
subplot(2,2,4)
plot(Xrnd(:,1),Xrnd(:,2),'k.','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
axis([-3.3 3.5 -7 7 ])
xlabel('\bf x1'); ylabel('\bf x2');
saveas(gcf,'MultiVariateTrv101','epsc')
save('MultiVariateTrv101','F')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 자유도가 $\nu = 5$이고 상관계수행렬이 다음과 같은 2변량 $\boldsymbol{t}_\nu$ 확률분포의 결합확률밀도함수, 이 결합확률밀도함수의 등고선도, 그리고 결합확률분포함수를 그린다.

$$\Sigma = \begin{bmatrix} 1 & 0.4 \\ 0.4 & 4 \end{bmatrix} \tag{1}$$

MATLAB함수들 mvtpdf, mvtcdf, 그리고 mvtrnd에서는 상관계수행렬을 크기행렬 Σ로 사용함에 유의하라. 이 MATLAB프로그램을 실행하면, 이 2변량 $\boldsymbol{t}$확률분포에서 100개 난수벡터들을 생성하고, 이 난수벡터들의 등고선도를 그린다. 이 그래프들이 그림 2.2.9에 그려져 있다. ■

예제 2.2.10 R을 이용해서 2변량 t확률분포의 확률질량함수, 등고선도, 확률분포함수, 그리고 t난수벡터를 살펴보기 위해서 다음 R프로그램 MultivariateTdistRV101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: MultivariateTdistRV101R.R
```

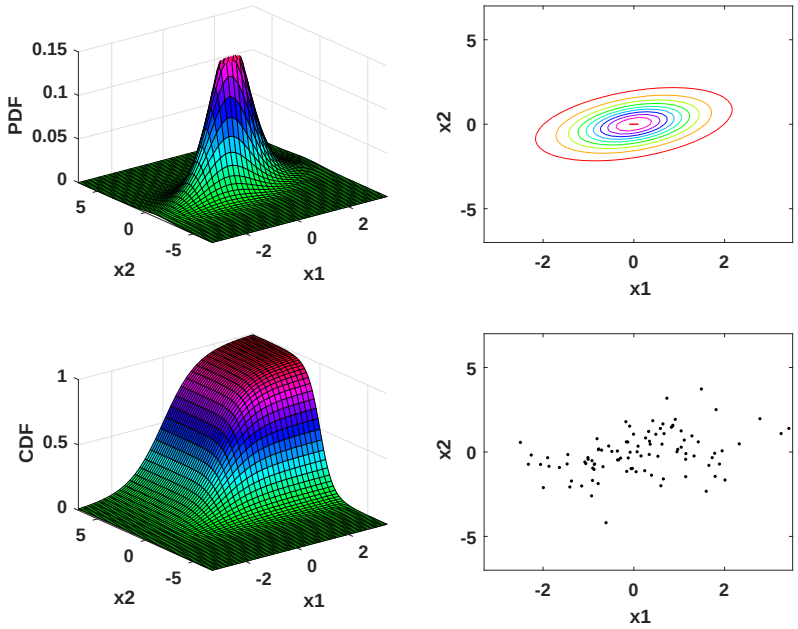


그림 2.2.9. 2변량 t 확률분포와 MATLAB

```
# Multivariate T Random Vector
# Programmed by CBS
# ---------------------------------------------------------------------------
# install.packages("mvtnorm")
library(mvtnorm)
( nu <- 5 )
( Sigma <- matrix(c(1,0.4,0.4,1),2,2) )

# Generating Normal random vectors
nSim <- 100
Xrand <- rmvt(nSim, Sigma, df=nu)

# Generate Density Function and its Contour
x1 <- seq(-2.5,3.0,length.out=21)
x2 <- seq(-5.0,5.5,length.out=21)
zT <- matrix(0,length(x1)*length(x2),ncol=1)
xT <- zT
yT <- zT
for (ii in 1:length(x1)) {
    for (jj in 1:length(x2)) {
        dum <- (ii-1)*length(x2)+jj
        xT[dum] <- x1[ii]
        yT[dum] <- x2[jj]
        xdum <- cbind(x1[ii],x2[jj])
        zT[dum] <- dmvt(x=xdum,sigma=Sigma,df=nu,log=FALSE)
    }
}

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('MultivariateTdistRV101R.eps')  # Start to save figure
```

```
RVdata1 <- data.frame(Xrand)
RVdata2 <- data.frame(xN,yN,zN)
plot11 <- ggplot(RVdata1, aes(x=Xrand[,1],y=Xrand[,2])) +
          geom_point(col="red",lwd=1.2) +
          xlab(expression(x_1)) + ylab(expression(x_2)) +
          geom_density2d()
plot12 <- ggplot(RVdata1, aes(x=Xrand[,1])) +
          geom_histogram(bins=20,fill="white",color="black")+
          xlab(expression(x_1))
plot13 <- ggplot(RVdata1, aes(x=Xrand[,2])) +
          geom_histogram(bins=20,fill="yellow",color="dark blue")+
          xlab(expression(x_2))
plot14 <- ggplot(RVdata2,aes(x=xT,y=yT,fill=zT)) +
          geom_tile() +
          geom_raster()
pushViewport(viewport(layout=grid.layout(2,2)))
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot13, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot14, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()            # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 자유도가 $\nu = 5$이고 상관계수행렬이 다음과 같은 2변량 $\boldsymbol{t}_\nu$ 확률분포의 결합확률밀도함수, 이 결합확률밀도함수의 등고선도, 그리고 결합확률분포함수를 그린다.

$$\Sigma = \begin{bmatrix} 1 & 0.4 \\ 0.4 & 4 \end{bmatrix} \tag{1}$$

R함수들 mvtpdf, mvtcdf, 그리고 mvtrnd에서는 상관계수행렬을 크기행렬 Σ로 사용함에 유의하라. 이 R프로그램을 실행하면, 이 2변량 $\boldsymbol{t}$확률분포에서 100개 난수벡터들을 생성하고, 이 난수벡터들의 등고선도를 그린다. 이 그래프들이 그림 2.2.10에 그려져 있다. ■

2.2.6 Wishart확률분포

Wishart확률분포는 다차원 카이제곱확률분포 또는 다차원 감마확률분포라고 할 수 있다. 이 소절에서는 Wishart확률분포의 정의와 난수발생기에 대해서 살펴보자. Wishart확률분포와 역Wishart확률분포는 베이지안통계학에서 중요한 역할을 한다.

확률벡터 $\boldsymbol{x}_i = [x_{i,1}, x_{i,2}, \cdots, x_{i,p}]^t$가 다음과 같은 확률분포를 따른다고 하자.

$$\boldsymbol{x}_i \overset{d}{\sim} \mathcal{N}_p(\boldsymbol{0}, \Sigma), \quad (i = 1, 2, \cdots, n) \tag{2.2.11}$$

이 확률벡터들 $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n$은 서로 독립이라고 가정하고, 다음과 같은 $p \times p$행렬 S를

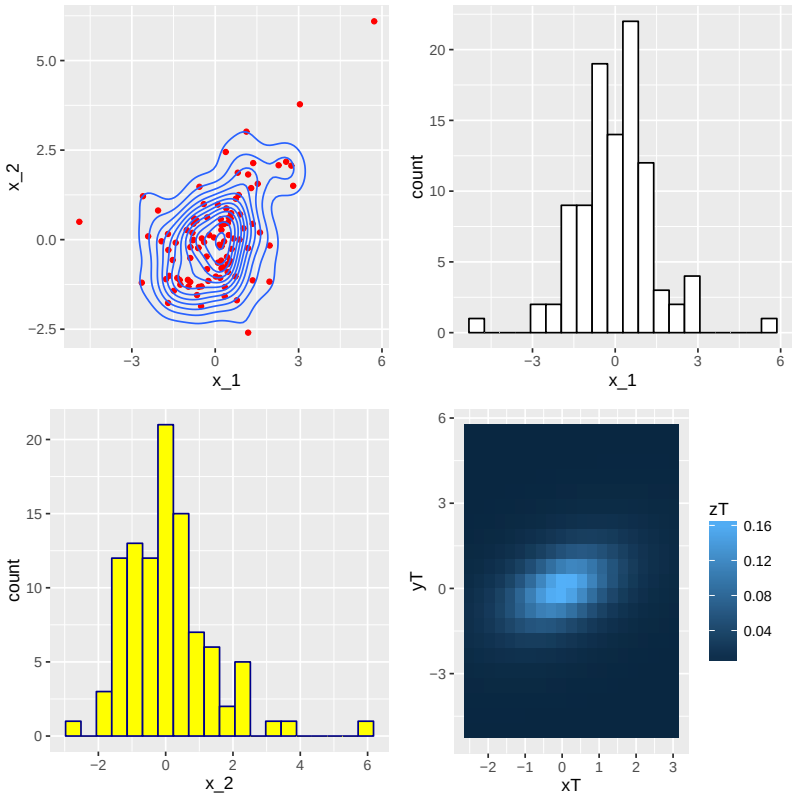


그림 2.2.10. 2변량 t확률분포와 R

정의하자.

$$S = \sum_{j=1}^{n} \boldsymbol{x}_j \boldsymbol{x}_j^t \tag{2.2.12}$$

이 S를 스캐터행렬(scatter matrix)이라 부른다. 이 확률행렬 S는 자유도가 n인 Wishart 확률분포를 따른다고 하고, 다음과 같이 표기하자.

$$S \overset{d}{\sim} \mathcal{W}_p(\Sigma, n) \tag{2.2.13}$$

만약 Σ가 가역이고, 또한 만약 $n \geq p$이면, 확률 1로(with probability 1) S도 가역이다. 만약 $p = 1$이고 $\Sigma = 1$이면, S는 자유도가 n인 카이제곱분포를 따른다.

만약 $n \geq p$이면, 확률행렬 S의 확률밀도함수는 다음과 같다.

$$f(S) = \frac{1}{2^{np/2}|\Sigma|^{n/2}\Gamma_p(\frac{n}{2})} |S|^{\frac{n-p-1}{2}} \exp\left(-\frac{1}{2} tr(\Sigma^{-1}S)\right) \tag{2.2.14}$$

여기서 $\Gamma_p(\cdot)$는 다음과 같이 정의되는 다변량 감마함수이다.

$$\Gamma_p\left(\frac{n}{2}\right) \doteq \pi^{p[p-1]/4} \prod_{j=1}^{p} \Gamma\left(\frac{n+1-j}{2}\right) \tag{2.2.15}$$

확률행렬 S의 평균행렬은 $n\Sigma$이고, 확률행렬 S의 (i,j) 원소인 $s_{i,j}$의 분산은 다음과 같다.

$$Var(s_{i,j}) = n[\sigma_{i,j}^2 + \sigma_{i,i}\sigma_{j,j}] \tag{2.2.16}$$

여기서 $\sigma_{i,j}$는 Σ의 (i,j) 원소이다.

예제 2.2.11 MATLAB을 이용해서 Wishart난수벡터를 생성하는 방법을 살펴보기 위해서 다음 MATLAB프로그램 WishartRV101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
% Filename: WishartRV101.m
% Inverse Wishart Random Vector
% Programmed by CBS
% ------------------------------------------------------------------------
clear, clf, close all
rand('twister',5489)
nSim = 2000
Sigma = [2 .5; .5 3]        % Covariance Matrix
D = chol(Sigma)             % Inverse of Covariance Matrix
for df=10:10:nSim
    n = df/10;
    Wish(:,:,n) = wishrnd(Sigma,df)/df;
end
nDisp = nSim/10;
nn = 1:nDisp;
dff =10*nn;
subplot(2,2,1)
dum11 = reshape(Wish(1,1,nn),1,nDisp);
plot(dff,dum11,'k.',[0 nSim],[Sigma(1,1),Sigma(1,1)],'r-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf df'); ylabel('\bf Wishart(1,1)');
subplot(2,2,2)
dum12 = reshape(Wish(1,2,nn),1,nDisp);
plot(dff,dum12,'k.',[0 nSim],[Sigma(1,2),Sigma(1,2)],'r-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf df'); ylabel('\bf Wishart(1,2)');
subplot(2,2,3)
dum21 = reshape(Wish(2,1,nn),1,nDisp);
plot(dff,dum21,'k.',[0 nSim],[Sigma(2,1),Sigma(2,1)],'r-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf df'); ylabel('\bf Wishart(2,1)');
subplot(2,2,4)
dum22 = reshape(Wish(2,2,nn),1,nDisp);
plot(dff,dum22,'k.',[0 nSim],[Sigma(2,2),Sigma(2,2)],'r-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf df'); ylabel('\bf Wishart(2,2)');
saveas(gcf,'WishartRV101','epsc')
save('WishartRV101','Wish')
```

```
% End of program
% --------------------------------------------------------------------------
```

Wishart확률분포에서 난수벡터를 생성하기 위해서, MATLAB함수 wishrnd를 사용할 수 있다.

이 MATLAB프로그램을 실행하면, 모수행렬 Σ가 다음과 같은 Wishart확률분포로부터 난수행렬을 생성한다.

$$\Sigma = \begin{bmatrix} 2 & 0.5 \\ 0.5 & 3 \end{bmatrix} \tag{1}$$

여기서 자유도는 10부터 390가지 10씩 증가하고, 각 자유도에 대해서 난수행렬이 1개 생성된다. 이렇게 생성된 난수들의 각 성분을 자유도로 나눈 값의 그래프가 그림 2.2.11에 그려져 있다. 그림 2.2.11의 좌측상단의 그래프는 $(1,1)$ 성분이고, 우측상단의 그래프는 $(1,2)$ 성분이고, 좌측하단의 그래프는 $(2,1)$ 성분이고, 우측하단의 그래프는 $(2,2)$ 성분이다. 이 그래프들에서 알 수 있듯이, 자유도 n이 증가하면 $\mathcal{W}_2(\Sigma, n)/n$이 Σ에 수렴한다. ■

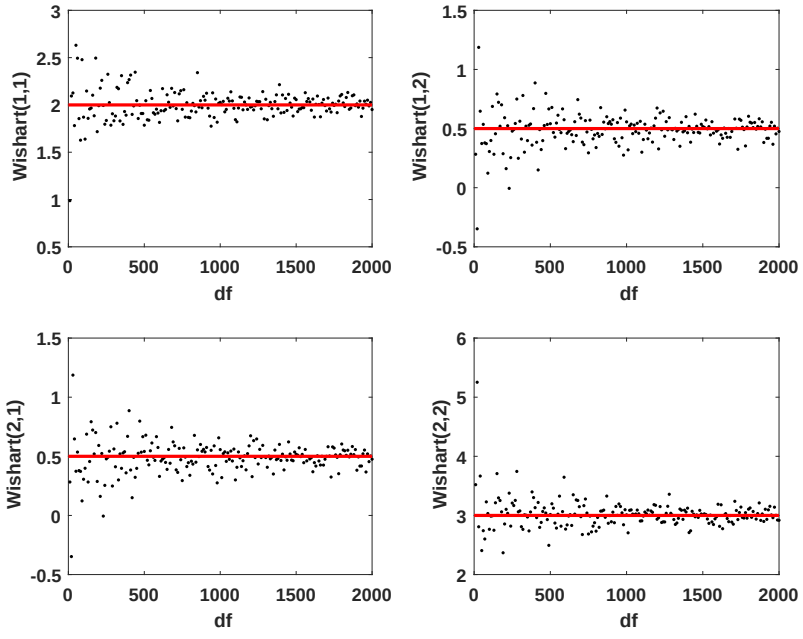


그림 2.2.11. 2변량 Wishart확률분포와 MATLAB

예제 2.2.12 R을 이용해서 Wishart난수벡터를 생성하는 방법을 살펴보기 위해서 다음 R 프로그램 WishartRV101R.R을 실행해 보자.

```
# --------------------------------------------------------------------------
# Filename: WishartRV101R.R
```

```
# Wishart Random Matrix
# Programmed by CBS
# --------------------------------------------------------------------------
# install.packages("MCMCpack")
library(MCMCpack)

# Making Data
set.seed(11)
p  <- 2
( Sigma <- matrix(c(2,0.5,0.5,3),2,2) )
nSim <- 2000
WW <- array(0,dim=c(p,p,nSim/10))

for (dff in 10:10:nSim) {
     ndf <- dff/10
     WW[,,ndf] <- rwish(dff,Sigma)/dff
}
nn <- 10*c(1:1:(nSim/10))
W11 <- WW[1,1,]
W12 <- WW[1,2,]
W21 <- WW[2,1,]
W22 <- WW[2,2,]

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('WishartRV101R.eps')  # Start to save figure
RVdata1 <- data.frame(nn,W11,W12,W21,W22)
plot11 <- ggplot(RVdata1,aes(x=nn,y=W11)) +
          geom_point(col="black",lwd=1.5) +
          xlab("df") + ylab("Wishart(1,1)") +
          geom_hline(yintercept=2,col="red",lwd=1)
plot12 <- ggplot(RVdata1,aes(x=nn,y=W12)) +
          geom_point(col="black",lwd=1.5) +
          xlab("df") + ylab("Wishart(1,2)") +
          geom_hline(yintercept=0.5,col="green",lwd=1)
plot21 <- ggplot(RVdata1,aes(x=nn,y=W12)) +
          geom_point(col="black",lwd=1.5) +
          xlab("df") + ylab("Wishart(2,1)") +
          geom_hline(yintercept=0.5,col="green",lwd=1)
plot22 <- ggplot(RVdata1,aes(x=nn,y=W22)) +
          geom_point(col="black",lwd=1.5) +
          xlab("df") + ylab("Wishart(2,2)") +
          geom_hline(yintercept=3,col="blue",lwd=1)
pushViewport(viewport(layout=grid.layout(2,2)))
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot21, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot22, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()            # End to save figure
# --------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 모수행렬 Σ가 다음과 같은 Wishart확률분포로부터 난수행렬을

생성한다.

$$\Sigma = \begin{bmatrix} 2 & 0.5 \\ 0.5 & 3 \end{bmatrix} \tag{1}$$

여기서 자유도는 10부터 390가지 10씩 증가하고, 각 자유도에 대해서 난수행렬이 1개 생성된다. 이렇게 생성된 난수들의 각 성분을 자유도로 나눈 값의 그래프가 그림 2.2.12에 그려져 있다. 그림 2.2.12의 좌측상단의 그래프는 $(1,1)$ 성분이고, 우측상단의 그래프는 $(1,2)$ 성분이고, 좌측하단의 그래프는 $(2,1)$ 성분이고, 우측하단의 그래프는 $(2,2)$ 성분이다. 이 그래프들에서 알 수 있듯이, 자유도 n이 증가하면 $\mathcal{W}_2(\Sigma, n)/n$이 Σ에 수렴한다. ■

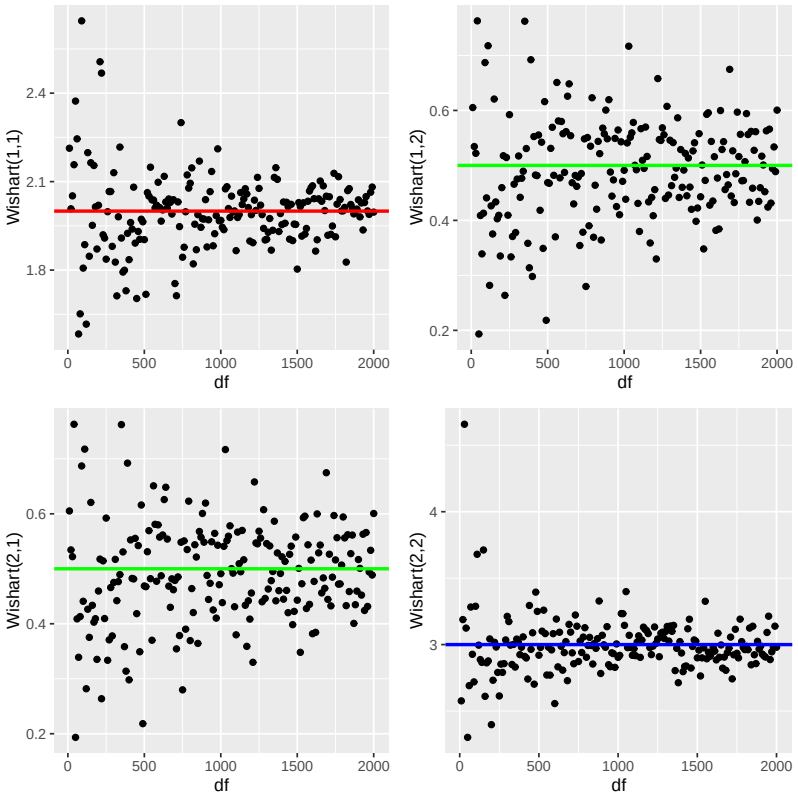


그림 2.2.12. 2변량 Wishart확률분포와 R

2.2.7 역Wishart확률분포

확률행렬 S가 다음과 같이 자유도가 n인 Wishart확률분포를 따른다고 하자.

$$S \overset{d}{\sim} \mathcal{W}_p(\Psi^{-1}, n) \tag{2.2.17}$$

확률행렬 $X = S^{-1}$는 자유도가 n인 역Wishart확률분포(inverse Wishart distribution 또는 inverted Wishart distribution)를 따른다고 하고, 다음과 같이 표기하자.

$$X \overset{d}{\sim} \mathcal{IW}_p(\Psi, n) \tag{2.2.18}$$

여기서 Ψ는 양정치행렬이고 또한 식 $n \geq p$이 성립한다고 가정하자. 역Wishart확률분포는 다변량정규확률분포의 켤레사전확률분포(conjugate prior probability distribution)이므로, 베이지안분석에서 자주 사용된다.

확률행렬 X의 확률밀도함수는 다음과 같다.

$$f(X) = \frac{|\Psi|^{n/2}}{2^{np/2}\Gamma_p(\frac{n}{2})}|X|^{-\frac{n+p+1}{2}} \exp\left(-\frac{1}{2}tr(\Psi X^{-1})\right) \tag{2.2.19}$$

여기서 $\Gamma_p(\cdot)$는 다변량 감마함수이다. 확률행렬 X의 평균행렬은 다음과 같다.

$$E(X) = \frac{1}{n-p-1}\Psi \tag{2.2.20}$$

확률행렬 X의 (i,j) 원소 $x_{i,j}$의 분산은 다음과 같다.

$$Var(x_{i,j}) = \frac{1}{[n-p][n-p-1]^2[n-p-3]}\left\{[n-p+1]\psi_{i,j}^2 + [n-p-1]\psi_{i,i}\psi_{j,j}\right\} \tag{2.2.21}$$

여기서 $\psi_{i,j}$는 Ψ의 (i,j) 원소이다. 식 (2.2.21)에서 알 수 있듯이, 다음 식이 성립한다.

$$Var(x_{i,i}) = \frac{2\psi_{i,i}^2}{[n-p-1]^2[n-p-3]} \tag{2.2.22}$$

또한, 공분산은 다음과 같다.

$$Cov(x_{i,j}, x_{k,l}) = \frac{1}{[n-p][n-p-1]^2[n-p-3]} \tag{2.2.23}$$

$$\cdot \{2\psi_{i,j}\psi_{k,l} + [n-p+1][\psi_{i,k}\psi_{j,l} + \psi_{i,l}\psi_{j,k}]\} \tag{2.2.24}$$

예제 2.2.13 MATLAB을 이용해서 역Wishart난수벡터를 생성하는 방법을 살펴보기 위해서 다음 MATLAB프로그램 iWishartRV101.m을 실행해 보자.

```
% ------------------------------------------------------------------
% Filename: iWishartRV101.m
```

```
% Inverse Wishart Random Vector
% Programmed by CBS
% ------------------------------------------------------------------
clear, clf, close all
rand('twister',5489)
Sigma = [2 .5; .5 3]        % Covariance Matrix
Psi = inv(Sigma)            % Inverse of Covariance Matrix
nSim = 2000
for df=10:10:nSim
    n = df/10;
    iWish(:,:,n) = iwishrnd(Psi,df)*df;
end
nDisp = nSim/10;
nn = 1:nDisp;
dff =10*nn;
subplot(2,2,1)
plot(dff,reshape(iWish(1,1,nn),1,nDisp),'k.', ...
    [0 nSim],[Psi(1,1),Psi(1,1)],'r-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf df'); ylabel('\bf iWishart(1,1)');
subplot(2,2,2)
plot(dff,reshape(iWish(1,2,nn),1,nDisp),'k.', ...
    [0 nSim],[Psi(1,2),Psi(1,2)],'r-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf df'); ylabel('\bf iWishart(1,2)');
subplot(2,2,3)
plot(dff,reshape(iWish(2,1,nn),1,nDisp),'k.', ...
    [0 nSim],[Psi(2,1),Psi(2,1)],'r-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf df'); ylabel('\bf iWishart(2,1)');
subplot(2,2,4)
plot(dff,reshape(iWish(2,2,nn),1,nDisp),'k.', ...
    [0 nSim],[Psi(2,2),Psi(2,2)],'r-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf df'); ylabel('\bf iWishart(2,2)');
saveas(gcf,'iWishartRV101','epsc')
save('iWishartRV101','iWish')
% End of program
% ------------------------------------------------------------------
```

역Wishart확률분포에서 난수벡터를 생성하기 위해서, MATLAB함수 iwishrnd를 사용할 수 있다.

이 MATLAB프로그램을 실행하면, 모수행렬 Ψ가 다음과 같은 역Wishart확률분포로부터 난수행렬을 생성한다.

$$\Psi = \begin{bmatrix} 2 & 0.5 \\ 0.5 & 3 \end{bmatrix}^{-1} = \begin{bmatrix} 0.5217 & -0.0870 \\ -0.0870 & 0.3478 \end{bmatrix} \tag{1}$$

여기서 자유도는 10부터 2000까지 10씩 증가하고, 각 자유도에 대해서 난수행렬이 1개 생성된다. 이렇게 생성된 난수들의 각 성분을 자유도로 나눈 값의 그래프가 그림 2.2.13에 그려져 있다. 그림 2.2.13의좌측상단의 그래프는 $(1,1)$ 성분이고, 우측상단의 그래프는 $(1,2)$ 성분이고,

좌측하단의 $(2,1)$ 그래프는 성분이고, 우측하단의 그래프는 $(2,2)$ 성분이다. 이 그래프들에서 알 수 있듯이, 자유도 n이 증가하면 $n\mathcal{IW}_2(\Psi, n)$이 Ψ에 수렴한다. ■

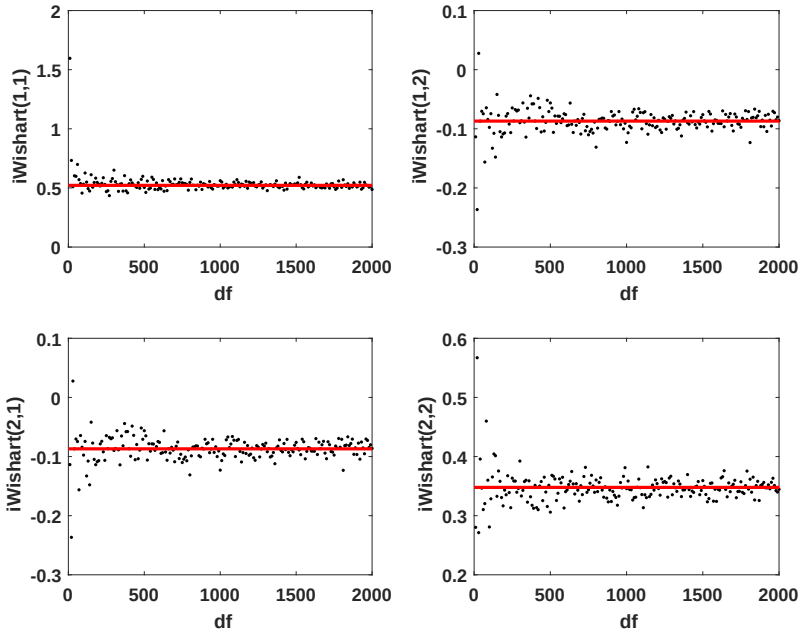


그림 2.2.13. 2변량 역Wishart확률분포와 MATLAB

예제 2.2.14 R을 이용해서 역Wishart난수벡터를 생성하는 방법을 살펴보기 위해서 다음 R 프로그램 iWishartRV101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------------
# Filename: iWishartRV101R.R
# Wishart Random Matrix
# Programmed by CBS
# ------------------------------------------------------------------------------
# install.packages("MCMCpack")
library(MCMCpack)

# Making Data
set.seed(11)
p  <- 2
( Sigma <- matrix(c(2,0.5,0.5,3),2,2) )
( Psi <- solve(Sigma) )
nSim <- 2000
iWW <- array(0,dim=c(p,p,nSim/10))

for (dff in 10:10:nSim) {
     ndf <- dff/10
     iWW[,,ndf] <- dff*riwish(dff,Psi)
}
nn <- 10*c(1:1:(nSim/10))
iW11 <- iWW[1,1,]
iW12 <- iWW[1,2,]
iW21 <- iWW[2,1,]
iW22 <- iWW[2,2,]
```

```
26
27 # Plotting
28 # install.packages("ggplot2")
29 library(ggplot2)
30 # install.packages("grid")
31 library(grid)
32 setEPS()
33 plot.new()
34 postscript('iWishartRV101R.eps')  # Start to save figure
35 RVdata1 <- data.frame(nn,iW11,iW12,iW21,iW22)
36 plot11 <- ggplot(RVdata1,aes(x=nn,y=iW11)) +
37           geom_point(col="black",lwd=1.5) +
38           xlab("df") + ylab("iWishart(1,1)") +
39           geom_hline(yintercept=0.5217,col="red",lwd=1)
40 plot12 <- ggplot(RVdata1,aes(x=nn,y=iW12)) +
41           geom_point(col="black",lwd=1.5) +
42           xlab("df") + ylab("iWishart(1,2)") +
43           geom_hline(yintercept=-0.0870,col="green",lwd=1)
44 plot21 <- ggplot(RVdata1,aes(x=nn,y=iW12)) +
45           geom_point(col="black",lwd=1.5) +
46           xlab("df") + ylab("iWishart(2,1)") +
47           geom_hline(yintercept=-0.0870,col="green",lwd=1)
48 plot22 <- ggplot(RVdata1,aes(x=nn,y=iW22)) +
49           geom_point(col="black",lwd=1.5) +
50           xlab("df") + ylab("iWishart(2,2)") +
51           geom_hline(yintercept=0.3478,col="blue",lwd=1)
52 pushViewport(viewport(layout=grid.layout(2,2)))
53 print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
54 print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
55 print(plot21, vp=viewport(layout.pos.row=2, layout.pos.col=1))
56 print(plot22, vp=viewport(layout.pos.row=2, layout.pos.col=2))
57 dev.off()            # End to save figure
58 # ------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 모수행렬 Ψ가 다음과 같은 역Wishart확률분포로부터 난수행렬을 생성한다.

$$\Psi = \begin{bmatrix} 2 & 0.5 \\ 0.5 & 3 \end{bmatrix}^{-1} = \begin{bmatrix} 0.5217 & -0.0870 \\ -0.0870 & 0.3478 \end{bmatrix} \tag{1}$$

여기서 자유도는 10부터 2000까지 10씩 증가하고, 각 자유도에 대해서 난수행렬이 1개 생성된다. 이렇게 생성된 난수들의 각 성분을 자유도로 나눈 값의 그래프가 그림 2.2.14에 그려져 있다. 그림 2.2.14의좌측상단의 그래프는 $(1,1)$성분이고, 우측상단의 그래프는 $(1,2)$성분이고, 좌측하단의 $(2,1)$그래프는 성분이고, 우측하단의 그래프는 $(2,2)$성분이다. 이 그래프들에서 알 수 있듯이, 자유도 n이 증가하면 $n\mathcal{IW}_2(\Psi,n)$이 Ψ에 수렴한다. ■

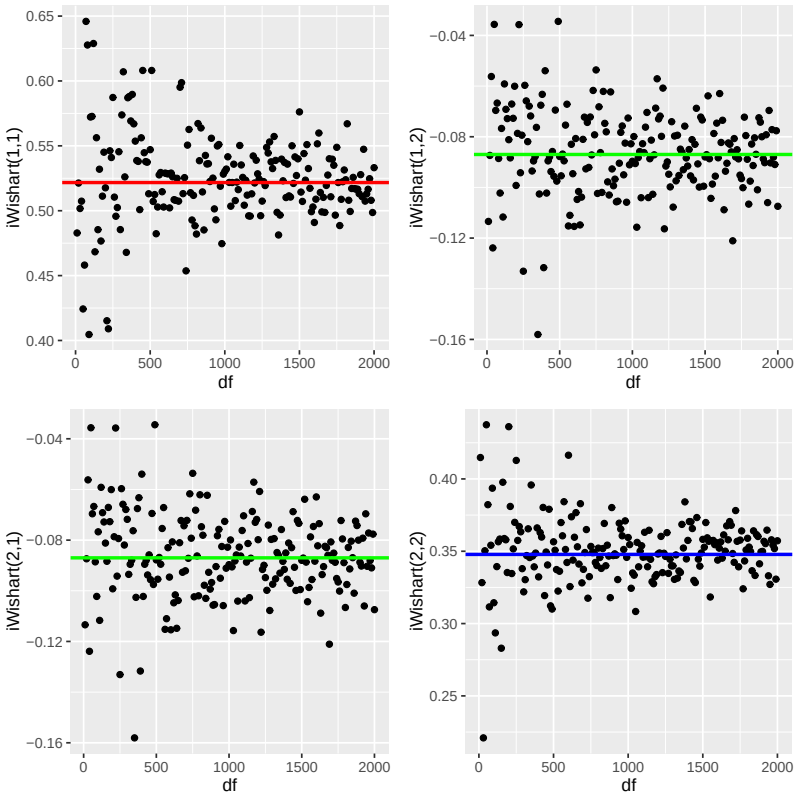


그림 2.2.14. 2변량 역Wishart확률분포와 R

제2.3절 혼합확률분포

확률분포함수들 $F_1, F_2, \cdots, F_n$ 인 확률벡터들 $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n$ 에 대해서, 다음과 같은 함수를 생각해보자.

$$F(\boldsymbol{x}) = \sum_{i=1}^{n} w_i F_i(\boldsymbol{x}) \tag{2.3.1}$$

여기서 $w_1, \cdots, w_n$ 은 다음 식들을 만족하는 가중값들이다.

$$w_i > 0, \quad (i = 1, 2, \cdots, n), \qquad \sum_{i=1}^{n} w_i = 1 \tag{2.3.2}$$

함수 $F(\boldsymbol{x})$ 도 확률분포함수이므로, 이에 해당하는 확률벡터를 $\boldsymbol{x}$ 로 표기하자. 만약 n 이 유한인 자연수이면, $F(\boldsymbol{x})$ 를 유한혼합확률분포함수(finite mixture probability distribution)라 부른다. 이에 해당하는 확률밀도함수(확률질량함수)는 다음과 같다.

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} w_i f_i(\boldsymbol{x}) \tag{2.3.3}$$

여기서 $f_1, f_2, \cdots, f_n$은 해당 확률밀도함수(확률질량함수)이다. 만약 확률벡터들 $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n$이 다변량정규확률분포들을 따르면, 확률벡터 $\boldsymbol{x}$를 Gauss혼합확률분포(Guass mixture probability distribution)이라 부른다.

확률벡터 $\boldsymbol{x}_i$의 평균벡터와 분산공분산행렬을 각각 $\boldsymbol{\mu}_i$와 Σ_i라고 하면, 다음 식들이 성립함을 알 수 있다.

$$\boldsymbol{\mu} \doteq E(\boldsymbol{x}) = \sum_{i=1}^{n} w_i \boldsymbol{\mu}_i \tag{2.3.4}$$

$$\Sigma \doteq Var(\boldsymbol{x}) = \sum_{i=1}^{n} w_i \left\{ \Sigma_i + [\boldsymbol{\mu}_i - \boldsymbol{\mu}][\boldsymbol{\mu}_i - \boldsymbol{\mu}]^t \right\} \tag{2.3.5}$$

만약 확률벡터 $\boldsymbol{x}_i$가 1차원이면, 즉 확률변수 x_i이면 다음 식들이 성립한다.

$$E([x-\mu]^j) = \sum_{i=1}^{n} \sum_{k=0}^{j} \binom{j}{k} [\mu_j - \mu]^{j-k} w_i E([x_i - \mu_i]^k), \quad (j = 3, 4, \cdots) \tag{2.3.6}$$

여기서 μ_i와 σ_i는 각각 확률변수 x의 평균과 분산이다.

혼합확률밀도함수는 간단한 확률밀도함수들의 선형결합으로 복잡한 확률분포를 표현한다. 각 성분확률밀도함수 $f_i(\boldsymbol{x})$가 각 하위 집합의 특성을 나타내므로, 혼합확률밀도함수는 전체 집합의 특성을 연구하는데 편리하다. 또한, 혼합확률밀도함수는 수학적으로 다루기 쉬우며, 특히 전체적으로 꼬리가 무거운 확률분포를 나타내는데 유용하다.

예제 2.3.1 일차원 Gauss혼합확률밀도함수를 그리는 예로서 다음 MATLAB프로그램 GaussMixtureConstruction101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: GaussMixtureConstruction101.m
%  Constructing a 1-D Gauss Mixture Distribution
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all
% Component PDF's
mu1 = -2; mu2 = 1;
MU = [ mu1 ; mu2 ]                                  % a column-vector.
sigma1 = 1; sigma2 = 1/2;  rho = 0.0;
Sigma(1,1,1) = sigma1^2; Sigma(1,1,2) = sigma2^2; % Variance not SD
weight = [ 0.25 ; 0.75 ]                            % a column-vector.
gmdd = gmdistribution(MU,Sigma,weight);
pdff = @(x)pdf(gmdd,x);
x = (-6:0.05:6)';                                    % a column-vector
f = pdff(x);
% Plotting
hold on
plot(x,weight(1,1)*normpdf(x,mu1,sigma1),'r-','linewidth',2)
plot(x,weight(2,1)*normpdf(x,mu2,sigma2),'g-','linewidth',2)
```

```
set(gca,'fontsize',11,'fontweigh','bold')
plot(x,f,'k.','linewidth',3)
legend('p_{1}*f_{1}(x)','p_{2}*f_{2}(x)','Mixture','location','NE')
xlabel('x'), ylabel('PDF')
hold off
saveas(gcf,'GaussMixtureConstruction101','epsc')
save('GaussMixtureConstruction101','f')
% End of program
% --------------------------------------------------------------------------
```

MATLAB함수 gmdistribution을 사용해서 Gauss혼합확률밀도함수값을 계산할 수 있다. 이 MATLAB프로그램을 실행하면, 다음과 같은 Gauss혼합확률밀도함수를 생성한다.

$$f(x) = \frac{1}{4} n\left(x \mid -2, 1^2\right) + \frac{3}{4} n\left(x \mid 1, 0.5^2\right) \tag{1}$$

이 Gauss혼합확률밀도함수의 그래프가 그림 2.3.1에 그려져 있다. 그림 2.3.1에서 적색 실선은 $\frac{1}{4} n\left(x \mid -2, 1^2\right)$을, 녹색 실선은 $\frac{3}{4} n\left(x \mid 1, 0.5^2\right)$을, 그리고 흑색 점선은 식 (1)의 Gauss 혼합확률밀도함수를 나타낸다. ■

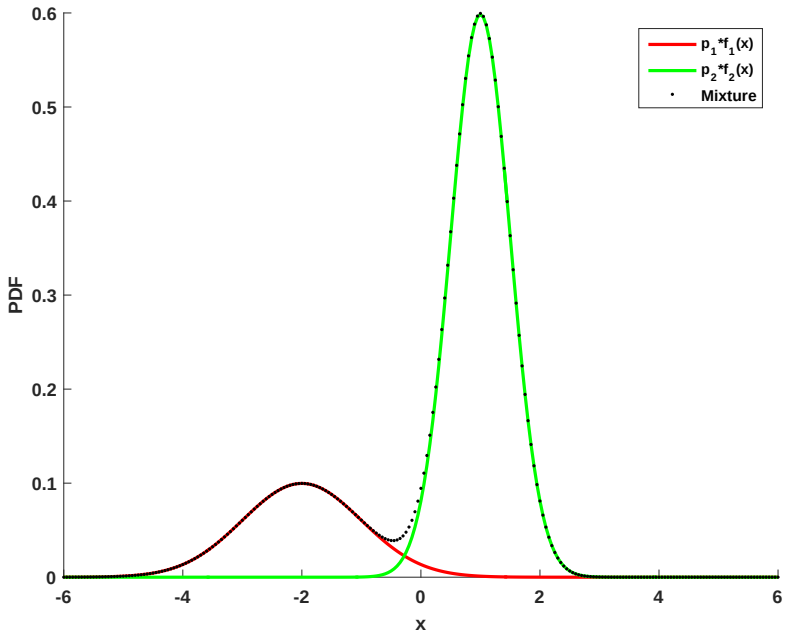


그림 2.3.1. 일차원 Gauss혼합확률밀도함수와 MATLAB

예제 2.3.2 일차원 Gauss혼합확률밀도함수를 그리는 예로서 다음 R프로그램 GaussMixtureConstruction101R.R을 실행해 보자.

```
# --------------------------------------------------------------------------
# Filename: GaussMixtureConstruction101R.R
# Gaussian Mixture (1D) Construction
```

```r
# Programmed by CBS
# ------------------------------------------------------------------------
x <- seq(-6, 6, length=301)
PD1 <-  1/4*dnorm(x+2)
PD2 <-  3/4*dnorm((x-1)*2)
PDmix <- PD1 + PD2
# Plotting
# install.packages("ggplot2")
library(ggplot2)
setEPS()
plot.new()
postscript('GaussMixtureConstruction101R.eps')  # Start to save figure
RVdata1 <- data.frame(x,PD1,PD2,PDmix)
ggplot(RVdata1) +
       geom_line(aes(x=x,y=PD1),col="red",lwd=1.2) +
       geom_line(aes(x=x,y=PD2),col="green",lwd=1.2) +
       geom_point(aes(x=x,y=PDmix),col="black",lwd=1.2)
dev.off()            # End to save figure
# ------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 다음과 같은 Gauss혼합확률밀도함수를 생성한다.

$$f(x) = \frac{1}{4} n\left(x \mid -2, 1^2\right) + \frac{3}{4} n\left(x \mid 1, 0.5^2\right) \tag{1}$$

이 Gauss혼합확률밀도함수의 그래프가 그림 2.3.2에 그려져 있다. 그림 2.3.2에서 적색 실선은 $\frac{1}{4} n\left(x \mid -2, 1^2\right)$을, 녹색 실선은 $\frac{3}{4} n\left(x \mid 1, 0.5^2\right)$을, 그리고 흑색 점선은 식 (1)의 Gauss 혼합확률밀도함수를 나타낸다. ■

예제 2.3.3 이차원 Gauss혼합확률밀도함수를 그리는 예로서 다음 MATLAB프로그램 GaussMixtureConstruction102.m을 실행해 보자.

```m
% ------------------------------------------------------------------------
%  Filename: GaussMixtureConstruction102.m
%  Constructing a 2-D Gauss Mixture Distribution
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all
mu1 = [ -1 -2 ]; mu2 = [2 1 ];
MU = [ mu1 ; mu2 ]
Sigma1 = [ 1 0 ; 0 1 ];
Sigma2 = [ 3 0 ; 0 1/2 ];
SIGMA = cat(3,Sigma1,Sigma2)                % Concatenate Matrices
weight = [ 1/3 ; 2/3 ]
obj = gmdistribution(MU,SIGMA,weight);
pdff = @(x,y)pdf(obj,[x y]);
x = (-8:0.1:8)'; y = x;
% Plotting
[ xx yy ] = meshgrid(x,y);
[sx1 sx2 ] = size(xx);
sizexy = sx1*sx2;
xr = reshape(xx,sizexy,1);
yr = reshape(yy,sizexy,1);
```

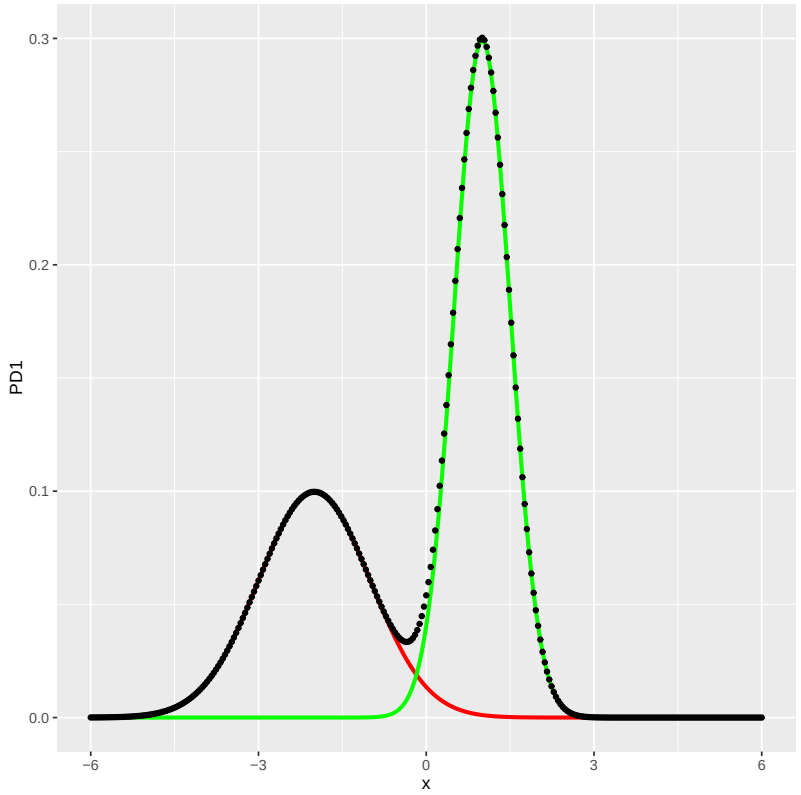


그림 2.3.2. 일차원 Gauss혼합확률밀도함수와 R

```
fr = pdff(xr,yr);
ff = reshape(fr,sx1,sx2);
mesh(xx,yy,ff)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('x_1'), ylabel('x_2')
zlabel('PDF')
axis([ -8 8 -8 8 0 0.1 ])
saveas(gcf,'GaussMixtureConstruction102','epsc')
save('GaussMixtureConstruction102','pdff')
% End of program
% --------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 다음과 같은 Gauss혼합확률밀도함수를 생성한다.

$$f(\boldsymbol{x}) = \frac{1}{3}n\left(\boldsymbol{x} \,\middle|\, \begin{bmatrix}-1\\-2\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) + \frac{2}{3}n\left(\boldsymbol{x} \,\middle|\, \begin{bmatrix}2\\1\end{bmatrix}, \begin{bmatrix}3 & 0\\0 & 1\end{bmatrix}\right) \tag{1}$$

이 Gauss혼합확률밀도함수의 그래프가 그림 2.3.3에 그려져 있다. ■

예제 2.3.4 이차원 Gauss혼합확률밀도함수를 그리는 예로서 다음 R프로그램 GaussMixtureConstruction102R.R을 실행해 보자.

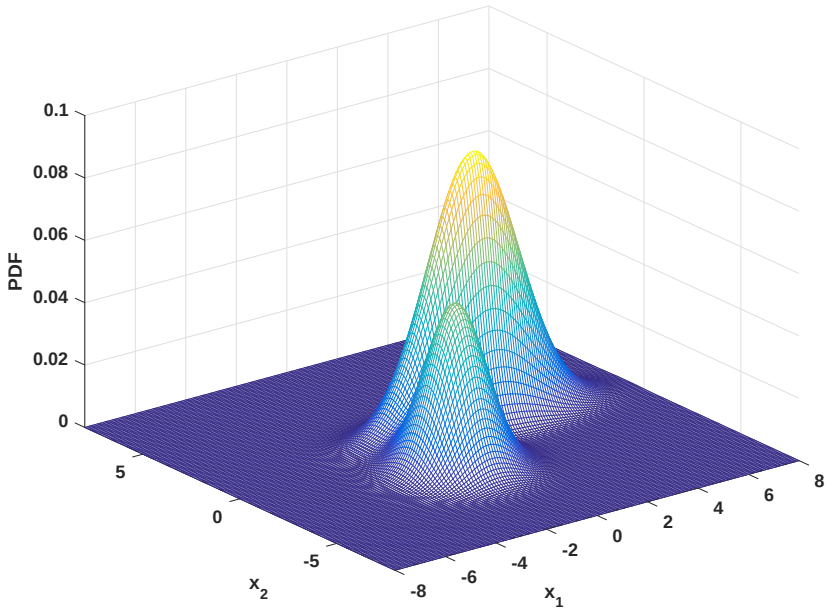


그림 2.3.3. 이차원 Gauss혼합확률밀도함수와 MATLAB

```r
# ------------------------------------------------------------------------------
# Filename: GaussMixtureConstruction102R.R
# Gaussian Mixture Construction (2D)
# Programmed by CBS
# ------------------------------------------------------------------------------
# install.packages("mvtnorm")
library(mvtnorm)
# Specify Parameters
( mu1 <- matrix(c(-1,-2),2,1))
( Sigma1 <- matrix(c(1,0,0,1),2,2))
( mu2 <- matrix(c(2,1),2,1))
( Sigma2 <- matrix(c(3,0,0,1),2,2))

set.seed(11)
nSim <- 10000
x1 <- matrix(0,nSim,1)
x2 <- matrix(0,nSim,1)
PDmix <- matrix(0,nSim,1)
for (ii in 1:nSim){
    x1[ii] <- runif(n=1, min=-6, max=6)
    x2[ii] <- runif(n=1, min=-6, max=6)
    xx <- cbind(x1[ii],x2[ii])
    PD1 <- dmvnorm(xx,mu1,Sigma1)
    PD2 <- dmvnorm(xx,mu2,Sigma2)
    PDmix[ii] <- 0.6665*PD1 + 0.3335*PD2
}

# Plotting
plot.new()
# install.packages("rgl")
require(rgl)
Xdata <- data.frame(x1,x2,PDmix)
plot3d(x1,x2,PDmix,type="s",size=0.3,lit=FALSE,col="green")
# rgl.postscript('GaussMixtureConstruction102R.eps',fmt='eps')
rgl.snapshot('GaussMixtureConstruction102R.png',fmt='png')
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 다음과 같은 Gauss혼합확률밀도함수에서 10000개 난수벡터들을 생성한다.

$$f(\boldsymbol{x}) = \frac{1}{3}n\left(\boldsymbol{x} \,\middle|\, \begin{bmatrix}-1\\-2\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) + \frac{2}{3}n\left(\boldsymbol{x} \,\middle|\, \begin{bmatrix}2\\1\end{bmatrix}, \begin{bmatrix}3 & 0\\0 & 1\end{bmatrix}\right) \tag{1}$$

이 난수벡터들의 3차원 산점도가 그림 2.3.4에 그려져 있다. 이 산점도는 식 (1)의 Gauss 혼합확률밀도함수를 나타낸다. ■

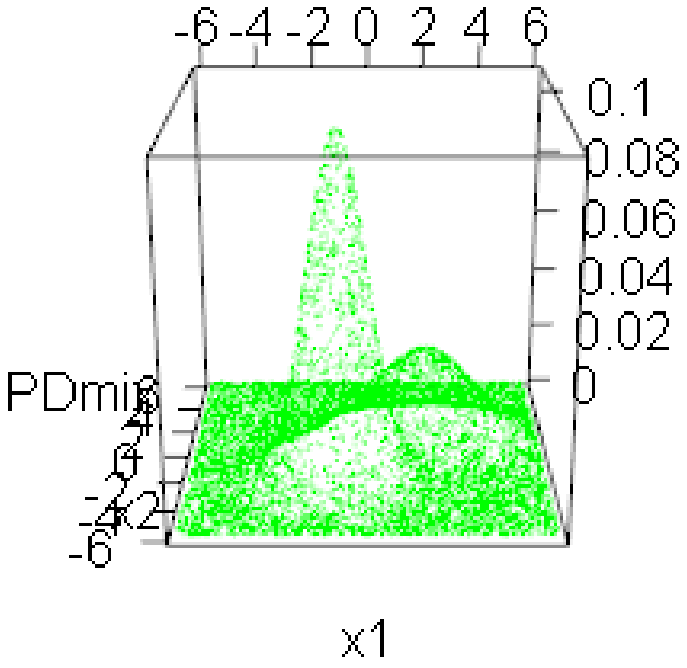


그림 2.3.4. 이차원 Gauss혼합확률밀도함수와 R

Gauss혼합확률밀도함수를 추정하는데는 일반적으로 EM알고리즘이 많이 사용된다. 이 내용을 자세히 다루는 것은 본서 수준을 넘는다. 이에 대한 자세한 내용은 본서의 후속인 *Bayesian Methods for Finance and Economics*에서 자세히 다루게 될 것이다. 여기서는 MATLAB함수 gmdistribution.fit를 사용해서 Gauss혼합확률밀도함수를 추정하는 예를 하나 살펴보기로 하자.

예제 2.3.5 Gauss혼합확률밀도함수를 추정하는 예로서 다음 MATLAB프로그램 GaussMixtureFit101.m을 실행해 보자.

```
% --------------------------------------------------------------------------
%  Filename: GaussMixtureFit101.m
%  Fitting a Mixture Distribution
%  Programmed by CBS
%--------------------------------------------------------------------------
clear all, close all
mu1 = [ -1 -2 ]; mu2 = [2 1 ];
Sigma1 = [ 1 0 ; 0 1 ]; Sigma2 = [ 3 0 ; 0 1/2 ];
rand('twister',5489)
X1 = mvnrnd(mu1,Sigma1,500);
X2 = mvnrnd(mu2,Sigma2,1000);
% Plotting Scatterplot
scatter(X1(:,1),X1(:,2),10,'b*');
set(gca,'fontsize',11,'fontweigh','bold')
hold on
scatter(X2(:,1),X2(:,2),10,'rd');
legend('X1','X2','location','NE')
xlabel('x_1'), ylabel('x_2')
% Fitting
X = [ X1; X2 ];
options = statset('disp','final');
obj = gmdistribution.fit(X,2,'Options',options);
ComponentNumber = obj.NComponents
ComponentWeight = obj.PComponents
ComponentMeans = obj.mu
ComponentCovariances = obj.Sigma
hold on
h = ezcontour(@(x,y)pdf(obj,[x y]),[-6 8],[-8 6]);
hold off
saveas(gcf,'GaussMixtureFit101','epsc')
save('GaussMixtureFit101','X1','X2')
% End of program
% --------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 2변량 정규확률분포 $\mathcal{N}\left(\begin{bmatrix}-1\\-2\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right)$ 에서 관찰벡터를 500개 그리고 2변량 정규확률분포 $\mathcal{N}\left(\begin{bmatrix}2\\1\end{bmatrix}, \begin{bmatrix}3 & 0\\0 & 1\end{bmatrix}\right)$ 에서 관찰벡터들 1000개를 생성한다. 이 관찰벡터들의 산포도가 그림 2.3.5에 그려져 있다. 이중에서 흑색 별표로 표시된 점들은 첫 번째 2변량 정규확률분포에서 출력된 것이고, 적색 다이아몬드로 표시된 점들은 두 번째 2변량 정규확률분포에서 출력된 것이다.

이 150개 관찰점들에 적합한 Gauss혼합확률밀도함수를 추정하기 위해서, MATLAB함수 gmdistribution.fit를 사용하였다. 이렇게 추정된 결과를 파일 obj에 저장하였다. 추정에 사용된 정규확률밀도들의 개수가 obj.Ncomponents에 기록되어 있다. 이 예제에서는 $n = 2$ 개의 정규확률밀도함수들이 사용되었다. 추정된 가중값들이 obj.Pcomponents에 기록되어 있다. 이 예제에서 가중값들은 $w_1 = 0.6665$ 이고 $w_2 = 0.3335$ 이다. 추정된 평균벡터들과 분산공분산행렬들이 각각 obj.mu와 obj.Sigma에 기록되어 있다. 이 예제에서 추정된 Gauss

혼합확률밀도함수는 다음과 같다.

$$f(\boldsymbol{x}) = 0.6665n\left(\boldsymbol{x}\,\middle|\,\begin{bmatrix}1.9947\\1.0452\end{bmatrix},\begin{bmatrix}2.9876 & -0.0497\\-0.0497 & 0.4939\end{bmatrix}\right) \tag{2.3.7}$$

$$+\,0.3335n\left(\boldsymbol{x}\,\middle|\,\begin{bmatrix}-0.9808\\-1.9365\end{bmatrix},\begin{bmatrix}0.9346 & 0.0466\\0.0466 & 1.0307\end{bmatrix}\right) \tag{1}$$

이렇게 추정된 Gauss혼합확률밀도함수의 등위곡선들이 그림 2.3.5에 그려져 있다. ■

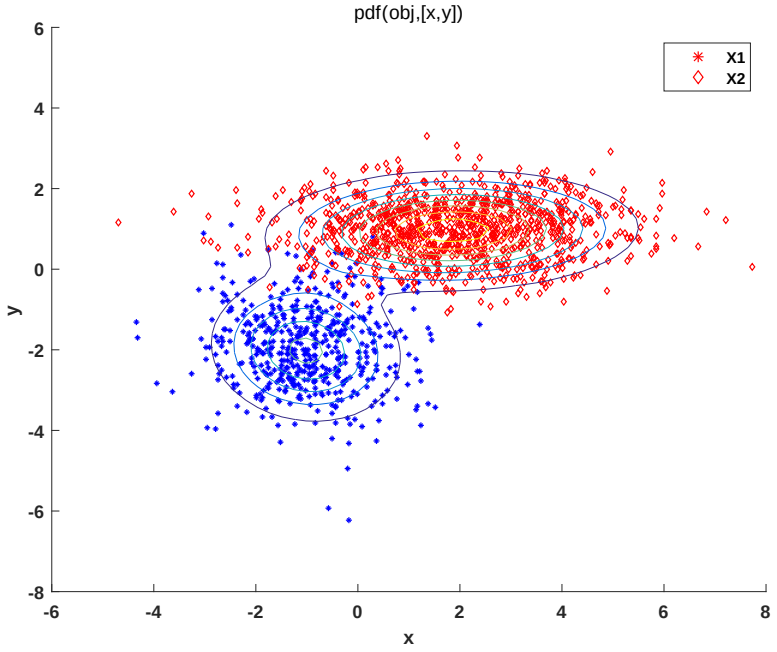


그림 2.3.5. Gauss혼합확률밀도함수의 추정

제2.4절 코푸라

> Essentially, all models are wrong, but some are useful.
>
> Box, G. E. P. and N. R. Draper (1987, p. 424)

각 $i(=1,2,\cdots,d)$에 대해서 연속형 확률변수 x_i의 확률분포함수를 $F_i = P(x_i \le x)$라고 하자. 여기서 확률변수들 $x_1, x_2, \cdots, x_d$는 서로 독립이 아니다. 다음 확률벡터를 정의하자.

$$[u_1, u_2, \cdots, u_d] \doteq [F_1(x_1), F_2(x_2), \cdots, F_d(x_d)] \tag{2.4.1}$$

이 확률벡터의 각 주변확률분포가 일양확률분포임은 명백하다. 확률벡터 $[x_1, x_2, \cdots, x_d]$의 코푸라(copula) C는 다음과 같은 확률벡터 $[u_1, u_2, \cdots, u_d]$의 결합확률분포함수이다.

$$C(a_1, a_2, \cdots, a_d) \doteq P(u_1 \leq a_1,\, u_2 \leq a_2,\, \cdots, u_d \leq a_d) \tag{2.4.2}$$

즉, 주변확률분포함수들 $F_1, F_2, \cdots, F_d$는 주변확률분포에 관한 모든 정보를 포함하고 있고, 코푸라 C는 확률변수들 $x_1, x_2, \cdots, x_d$사이의 상관관계에 관한 모든 정보를 포함하고 있다. 즉, 코푸라란 단변량확률분포들을 연결해서 다변량확률분포를 만드는 함수를 뜻한다.

코푸라가 중요한 이유는 코푸라를 사용해서 일반적인 다변량확률분포함수에서 난수벡터를 생성할 수 있다는 것이다. 즉, 서로 상관관계를 갖는 일양확률변수들의 벡터 $[u_1, u_2, \cdots, u_d]$로부터 난수벡터를 생성한 다음, 다음 식을 이용해서 다른 다변량확률분포함수에서 난수벡터 $[x_1, x_2, \cdots, x_d]$를 생성한다.

$$[x_1, x_2, \cdots, x_d] \doteq [F_1^{-1}(u_1), F_2^{-1}(u_2), \cdots, F_d^{-1}(u_d)] \tag{2.4.3}$$

우선 2차원 코푸라를 살펴보면, 2차원 코푸라는 다음 식들을 만족하는 함수 $C : [0,1]^2 \to [0,1]$이다.

$$C(0,t) = C(t,0) = 0, \quad (0 \leq t \leq 1) \tag{2.4.4}$$

$$C(1,t) = C(t,1) = t, \quad (0 \leq t \leq 1) \tag{2.4.5}$$

$$C(u_2, v_2) - C(u_1, v_2) - C(u_2, v_1) + C(u_1, v_1) \geq 0,$$
$$(0 \leq u_1 \leq u_2 \leq 1,\ 0 \leq v_1 \leq v_2 \leq 1) \tag{2.4.6}$$

정의 2.4.1: 코푸라

만약 어떤 결합확률분포의 각 주변확률분포가 구간 $(0,1)$ 상에서 일양확률분포이면, 이 결합확률분포함수를 코푸라라 한다.

예제 2.4.1 다음과 같은 확률벡터 (u, v)의 결합확률분포함수들을 살펴보자.

$$M(u,v) = \min\{u,v\} 1_{[0,1]\times[0,1]}(u,v) \tag{1}$$

$$W(u,v) = \max\{u+v-1, 0\}\, 1_{[0,1]\times[0,1]}(u,v) \tag{2}$$

$$\Pi(u,v) = uv\, 1_{[0,1][0,1]}(u,v) \tag{3}$$

식 (1) ∼식 (3)의 각 결합확률분포함수가 코푸라임은 명백하다. Fréchet-Hoeffding의 상한 M은 직선 $v = u$상에 확률이 모여 있는 확률분포함수이며, Fréchet-Hoeffding의 하한 W는 직선 $v = 1 - u$상에 확률이 모여 있는 확률분포함수이고, Π는 u와 v가 서로 독립인 확률분포함수이다. 임의의 코푸라 C에 대해서 다음 식들이 성립한다.

$$W(u,v) \le C(u,v) \le M(u,v) \tag{4}$$

즉, M과 W는 각각 코푸라의 상한과 하한이다. ■

다음과 같은 Sklar정리에 의하면, 일반적인 결합확률분포함수를 코푸라를 사용해서 표현할 수 있다.

정리 2.4.1: Sklar정리

확률벡터 $[x_1, x_2]^t$의 결합확률분포함수를 $H(x_1, x_2)$라 하고 또한 그 주변확률분포함수들을 각각 $F(x_1)$과 $G(x_2)$라 하면, 다음 식을 만족하는 코푸라 C가 존재한다.

$$H(x_1, x_2) = C(F(x_1), G(x_2)), \quad (-\infty < x_1, x_2 < \infty) \quad (*)$$

또한, 만약 F와 G가 연속이라면, 코푸라 C가 일의적으로 정해진다. 반대로, 만약 C가 코푸라이고 F와 G를 단변량 확률분포함수들이라 하면, 식 (*)로 정의되는 H는 F와 G를 주변확률분포함수들로 하는 결합확률분포함수이다.

이 Sklar정리에 의하면, 코푸라와 주변확률분포들이 정해지면 결합확률분포가 결정된다. 즉, 코푸라를 사용하면, 각 결합확률분포에 대해 주변확률분포들과 상관성을 분리해서 생각할 수 있다.

보조정리 2.4.1

확률벡터 $[x_1, x_2]^t$의 결합확률분포함수를 $H(x_1, x_2)$라 하고 또한 그 주변확률분포함수들 $F(x_1)$과 $G(x_2)$가 연속이라고 하면, 다음 식을 만족하는 코푸라 C가 존재한다.

$$C(u, v) = H(F^{-1}(u), G^{-1}(v)), \quad (0 < u, v < 1)$$

높은 차원의 코푸라를 다음과 같이 정의하자.

정의 2.4.2

다음 성질들을 모두 만족하는 $C : [0,1]^d \to [0,1]$을 d차원 코푸라라 한다.

1) 각 $\boldsymbol{x} = [x_1, \cdots, x_k, \cdots, x_d] (\in [0,1]^d)$에 대해서 다음 성질들이 성립한다. 첫째, 만약 적어도 하나의 k에 관해 $x_k = 0$이면, 식 $C(\boldsymbol{x}) = 0$이 성립한다. 둘째, 임의의 k와 임의의 $x_k (\in [0,1])$에 대해서, 다음 식이 성립한다.

$$C(1, \cdots, 1, x_k, 1, \cdots, 1) = x_k$$

2) 집합 $B = \prod_{k=1}^{d} [a_k, b_k]$가 모든 정점들이 $[0,1]^d$에 포함되는 d차원 입방체라 하자. 임의의 집합 B에 대해서 다음 식이 성립한다.

$$\sum_{u} sgd(\boldsymbol{x}) C(\boldsymbol{u}) \geq 0$$

여기서 $\boldsymbol{u}$는 B의 정점이며, $\sum_u$는 모든 정점들에 대한 합이다. 또한, 식 $u_k = a_k$을 만족하는 k가 짝수 개이면 $sgd(\boldsymbol{u}) = 1$이고, 식 $u_k = a_k$을 만족하는 k가 홀수 개이면 $sgn(\boldsymbol{u}) = -1$이다.

높은 차원의 코푸라에 대해서도 Sklar정리가 성립한다.

정리 2.4.2

어떤 d차원 확률벡터의 결합확률분포함수를 H, 그리고 그 주변확률분포함수들을 $F_1, F_2, \cdots, F_d$이라 하자. 각 $\boldsymbol{x} \left(\in (-\infty, \infty)^d\right)$에 대해서 다음 식을 만족시키는 코푸라 C

가 존재한다.

$$H(\boldsymbol{x}) = C(F_1(x_1), F_2(x_2), \cdots, F_d(x_d)) \quad (**)$$

만약 $F_1, F_2, \cdots, F_d$이 모두 연속이라면, 코푸라 C는 일의적으로 존재한다. 반대로, 만약 C가 d차원 코푸라이고 또한 $F_1, F_2, \cdots, F_d$가 단변량 확률분포함수들이면, 식 (**)로 정의되는 H는 $F_1, F_2, \cdots, F_d$을 주변확률분포함수들로 하는 d차원 결합확률분포함수이다.

예제 2.4.2 평균벡터가 $\mathbf{0}$이고 분산공분산행렬이 주어진 상관계수행렬 P인 d변량 정규확률벡터의 결합확률분포함수를 N_p라 하고, 표준정규확률분포의 확률분포함수를 N이라 하자. 모수행렬이 P인 Gauss코푸라 C_p^{Gauss}는 다음과 같이 정의된다.

$$C_p^{Gauss}(u_1, u_2, \cdots, u_d) \doteq N_p(N^{-1}(u_1), N^{-1}(u_2), \cdots, N^{-1}(u_d)) \tag{1}$$

평균벡터가 $\mathbf{0}$이고 분산공분산행렬이 주어진 상관계수행렬 P인 d 확률벡터의 결합확률분포함수를 $t_{\nu,P}$라 하고, 자유도가 ν인 t확률분포의 확률분포함수를 t_ν이라 하자. 모수행렬이 P인 t코푸라 $C_p^{Student}$는 다음과 같이 정의된다.

$$C_P^{Student}(u_1, u_2, \cdots, u_d) \doteq t_{\nu,P}(t_\nu^{-1}(u_1), t_\nu^{-1}(u_2), \cdots, t_\nu^{-1}(u_d)) \tag{2}$$

■

예제 2.4.3 MATLAB을 사용해서 코푸라를 계산하는 예로서 다음 MATLAB프로그램 Copula_t_CDF101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: Copula_t_CDF101.m
%  t Copula CDF
%  Programmed by CBS
% ------------------------------------------------------------------------
clear all, close all
nint = 31;
u = linspace(0,1,nint);
[U1,U2 ] = meshgrid(u,u);
% t Copula
Fdum = copulacdf('t',[U1(:) U2(:)],0.5,11);
Fc = reshape(Fdum,nint,nint);
surf(U1,U2,Fc)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf u_{1}','fontsize',12);
ylabel('\bf u_{2}','fontsize',12);
saveas(gcf,'Copula_t_CDF101','epsc')
save('Copula_t_CDF101','Fdum','Fc')
```

```
% End of program
% --------------------------------------------------------------------------
```

MATLAB함수 copulacdf와 copulapdf를 사용해서 Gauss코푸라와 t코푸라와 그에 해당하는 확률밀도함수들을 계산할 수 있다.

Gauss코푸라와 t코푸라를 계산하기 위한 MATLAB함수 couplacdf의 사용법은 각각 다음과 같다.

```
YG = copulacdf('Gaussian',U,rho)
Yt = copulacdf('t',U,rho,df)
```

여기서 n개의 d차원 Gauss코푸라 YG를 계산하기 위한 입력변수 U는 $n \times d$행렬이고 입력변수 rho는 상관계수행렬 또는 상수인 상관계수이다. 또한, t코푸라 Yt를 계산하기 위해서는 자유도 df를 추가로 입력해야 한다.

이 MATLAB프로그램을 실행하면, 자유도가 11이고 상관계수가 0.5인 2차원 t코푸라로부터 31^2개 벡터들을 계산한다. 이 벡터들의 그래프가 그림 2.4.1에 그려져 있다. ■

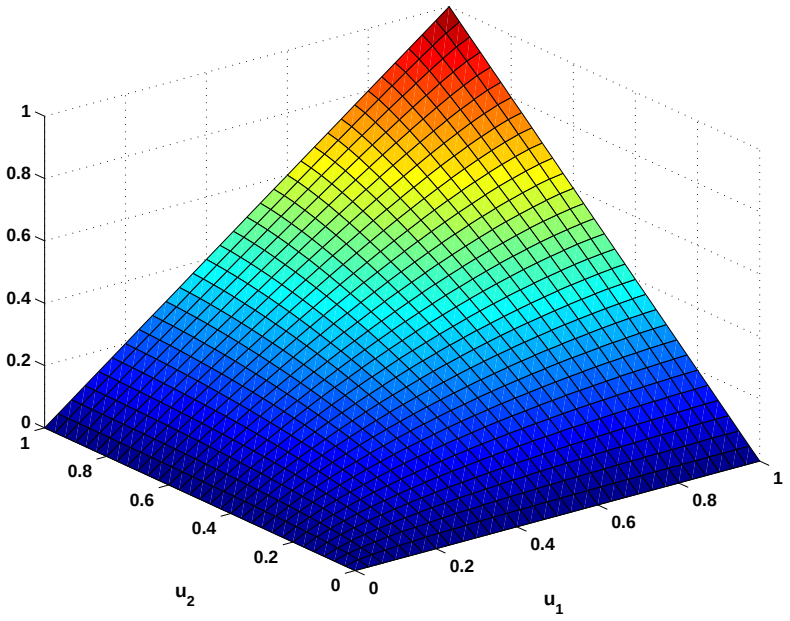


그림 2.4.1. t 코푸라와 MATLAB

예제 2.4.4 R을 사용해서 코푸라를 계산하는 예로서 다음 R프로그램 Copula_t_CDF101R.m을 실행해 보자.

```
# --------------------------------------------------------------------------
# Filename: Copula_t_CDF101R.R
```

```
# Gaussian and t Copulas
# Programmed by CBS
# --------------------------------------------------------------------------
# install.packages("copula")
library("copula")
set.seed(11)
nSim <- 100
# myCopula.norm <- ellipCopula(family = "normal", dim = 3, dispstr = "ex",
#                              param = 0.5)
# uRand <- rcopula(myCopula.norm, nSim)
myCopula.t <- ellipCopula(family = "t", dim = 3, dispstr = "toep",
                          param = c(0.6, -0.2), df = 11)
uRand <- rcopula(myCopula.t, nSim)
normuR <- matrix(0,nSim,1)
for (ii in 1:nSim){
dum <- uRand[ii,]
normuR[ii] <- norm(dum,"2")
}

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('Copula_t_CDF101R.eps')  # Start to save figure
RVdata1 <- data.frame(uRand,normuR)
plot11 <- ggplot(RVdata1, aes(x=uRand[,1],y=uRand[,2])) +
          geom_point(col="red",lwd=1.2) +
          xlab(expression(u_1)) + ylab(expression(u_2)) +
          geom_density2d()
plot13 <- ggplot(RVdata1, aes(x=uRand[,1],y=uRand[,3])) +
          geom_point(col="green",lwd=1.2) +
          xlab(expression(u_1)) + ylab(expression(u_3)) +
          geom_density2d()
plot23 <- ggplot(RVdata1, aes(x=uRand[,2],y=uRand[,3])) +
          geom_point(col="blue",lwd=1.2) +
          xlab(expression(u_2)) + ylab(expression(u_3)) +
          geom_density2d()
plot44 <- ggplot(RVdata1, aes(x=normuR)) +
          geom_histogram(bins=20,fill="white",color="black")+
          xlab("Norm of u")
pushViewport(viewport(layout=grid.layout(2,2)))
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot13, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot23, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot44, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()            # End to save figure
# --------------------------------------------------------------------------
```

R함수 copulacdf와 copulapdf를 사용해서 Gauss코푸라와 t코푸라와 그에 해당하는 확률밀도함수들을 계산할 수 있다.

Gauss코푸라와 t코푸라의 목적함수들을 지정하기 R함수 ellipCoupla의 사용법은 각각 다음과 같다.

```
CopulaObj <- ellipCopula(family="normal", dim=d, dispstr="type",param=rho1)
```

```
CopulaObj <- ellipCopula(family="t", dim=d, dispstr="type",
                         param=c(rho1,rho2), df=m)
```

즉, Gauss코푸라를 계산하기 위해서는 입력모수 family에 `"normal"`를 지정하고, 상관행렬의 차수 d를 입력모수 `dim`에 할당하고, 상관행렬의 형태(type)를 나타내는 입력모수 dispstr에는 `"ar1"`, `"ex"`, `"toep"`, `"un"`을 지정할 수 있다. 예를 들어, 차수가 `dim=3`인 경우에 해당하는 상관행렬들은 다음과 같다.

$$R_{ar1} = \begin{bmatrix} 1 & \rho_1 & \rho_1^2 \\ \rho_1 & 1 & \rho_1 \\ \rho_1^2 & \rho_1 & 1 \end{bmatrix}, \qquad R_{ex} = \begin{bmatrix} 1 & \rho_1 & \rho_1 \\ \rho_1 & 1 & \rho_1 \\ \rho_1 & \rho_1 & 1 \end{bmatrix}, \tag{1}$$

$$R_{toep} = \begin{bmatrix} 1 & \rho_1 & \rho_2 \\ \rho_1 & 1 & \rho_1 \\ \rho_2 & \rho_1 & 1 \end{bmatrix}, \qquad R_{ex} = \begin{bmatrix} 1 & \rho_1 & \rho_2 \\ \rho_1 & 1 & \rho_3 \\ \rho_2 & \rho_3 & 1 \end{bmatrix} \tag{2}$$

또한, 입력모수 param에는 해당되는 상관계수들을 할당한다. 예를 들어, 상관행렬의 형태가 ar1이면 입력모수 param에 $c(\rho_1)$을, 상관행렬의 형태가 ex이면 입력모수 param에 $c(\rho_1)$을, 상관행렬의 형태가 toep이면 입력모수 param에 $c(\rho_1, \rho_2)$를, 상관행렬의 형태가 un이면 입력모수 param에 $c(\rho_1, \rho_1, \rho_3)$를 입력한다. 여기서 ar1은 $AR(1)$모형의 상관행렬을, ex는 교환가능한(exchangeable) 상관행렬을, toep는 Toeplitz행렬을, 그리고 un은 특별한 구조가 없는 상관행렬을 상징한다. 이렇게 정해진 목적함수에서 난수벡터들을 생성하기 위해서는 R명령문 `rcopula(CopulaObj,nSim)`을 사용한다. 여기서 nSim은 생성할 난수벡터들의 개수이다.

이 R프로그램을 실행하면, 자유도가 11이고 상관행렬이 다음과 같은 0.5인 3차원 t 코푸라로부터 100개 난수벡터들을 생성한다.

$$R_{toep} = \begin{bmatrix} 1 & 0.6 & -0.2 \\ 0.6 & 1 & 0.6 \\ -0.2 & 0.6 & 1 \end{bmatrix} \tag{3}$$

이 난수벡터들의 상관계수들과 난수벡터의 크기(스펙트럴노름)의 히스토그램이 그림 2.4.2에 그려져 있다. ■

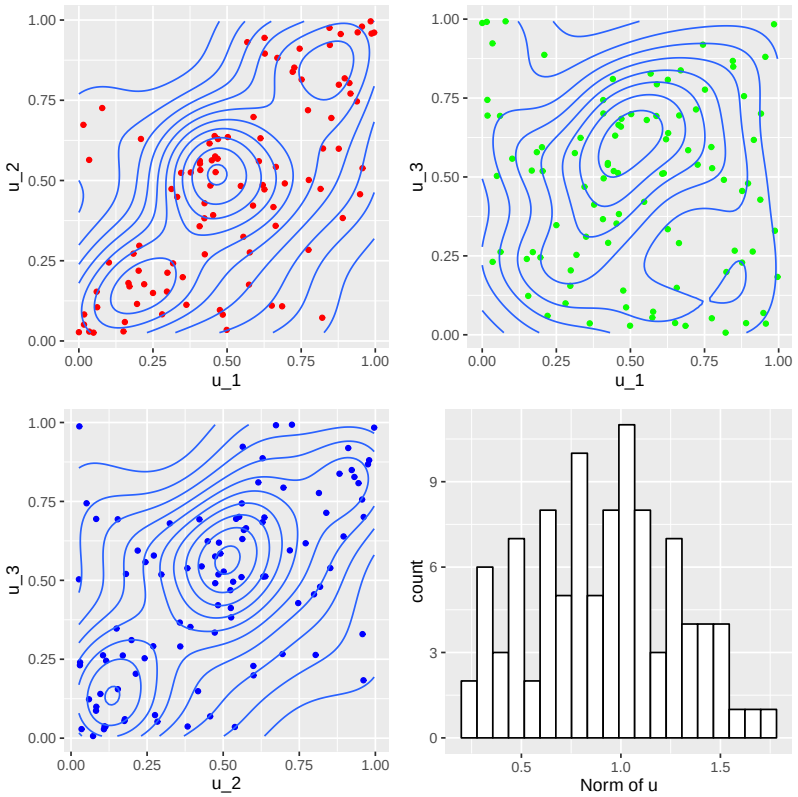


그림 2.4.2. t 코푸라와 R

함수 $C : [0,1]^d \to [0,1]$를 다음과 같이 정의하자.

$$C(\boldsymbol{x}) \doteq \psi(\psi^{-1}(x_1) + \psi^{-1}(x_2) + \cdots + \psi^{-1}(x_d)) \tag{2.4.7}$$

식 (2.4.7)의 ψ를 생성자(generator)라 부른다. 함수 C가 d차원 코푸라가 되기 위한 필요충분조건은 ψ가 다음 두 조건들을 만족하는 것이다. 첫째, 생성자 ψ가 임의의 $x(\in [0,\infty))$에 관해 다음 식들을 만족한다.

$$[-1]^k \frac{d^{(k)}\psi(x)}{dx^k} \geq 0, \quad (k = 0, 1, \cdots, d-2) \tag{2.4.8}$$

둘째, $[-1]^{d-1}\psi^{(d-2)}(x)$가 비증가이며 아래로 볼록이다. 이러한 두 조건들을 만족하는 함수는 $d-$단조(d-monotone)라고 한다. 이러한 d차원 코푸라 C를 Alchimedes형 코푸라라 한다. Archimedean코푸라의 특징은 결합확률분포함수가 하나의 함수 C로 기술할 수 있다는 것이다. 대표적인 Archimedean코푸라로는 Clayton코푸라, Frank코푸라, Gumbel코푸라 등이 있다. 각 코푸라의 생성자는 다음과 같다.

$$\text{Clayton}: \quad \psi(t) = t^{-\alpha} - 1, \quad (\alpha \in (0, \infty)) \tag{2.4.9}$$

$$\text{Frank}: \quad \psi(t) = -\ln \frac{\exp(-\alpha t)-1}{\exp(-\alpha)-1}, \quad (\alpha \in (0, \infty)) \tag{2.4.10}$$

$$\text{Gumbel}: \quad \psi(t) = [-\ln t]^{\alpha}, \quad (\alpha \in [1, \infty)) \tag{2.4.11}$$

이 생성자들에 대해서는 Yan [45]을 참조하라.

예제 2.4.5 MATLAB을 사용해서 Archimedean코푸라들을 계산하는 예로서 다음 MATLAB 프로그램 CopulaCDFs101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: CopulaCDFs101.m
%  Copula CDF's
%  Programmed by CBS
% ------------------------------------------------------------------------
clear all, close all
nint = 11;
u = linspace(0,1,nint);
[U1,U2 ] =meshgrid(u,u);
% Clayton Copula
Fdum = copulacdf('Clayton',[U1(:) U2(:)],2);
Fc = reshape(Fdum,nint,nint);
subplot(2,2,1)
contour(U1,U2,Fc)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf u_{1}','fontsize',12);
ylabel('\bf u_{2}','rotation',0,'fontsize',12);
title('\bf Clayton copula','fontsize',12)
% Frank Copula
Fdum = copulacdf('Frank',[U1(:) U2(:)],2);
Fc = reshape(Fdum,nint,nint);
subplot(2,2,2)
contour(U1,U2,Fc)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf u_{1}','fontsize',12);
ylabel('\bf u_{2}','rotation',0,'fontsize',12);
title('\bf Frank copula','fontsize',12)
% Gumbel Copula
Fdum = copulacdf('Gumbel',[U1(:) U2(:)],2);
Fc = reshape(Fdum,nint,nint);
subplot(2,2,3)
contour(U1,U2,Fc)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf u_{1}','fontsize',12);
ylabel('\bf u_{2}','rotation',0,'fontsize',12);
title('\bf Gumbel copula','fontsize',12)
% Gaussian Copula
Fdum = copulacdf('Gaussian',[U1(:) U2(:)],0.5);
Fc = reshape(Fdum,nint,nint);
subplot(2,2,4)
contour(U1,U2,Fc)
set(gca,'fontsize',11,'fontweigh','bold')
```

```
xlabel('\bf u_{1}','fontsize',12);
ylabel('\bf u_{2}','rotation',0,'fontsize',12);
title('\bf Gaussian copula','fontsize',12)
saveas(gcf,'CopulaCDFs101','epsc')
save('CopulaCDFs101','Fdum','Fc')
% End of program
% ------------------------------------------------------------------------
```

MATLAB함수 copulacdf와 copulapdf를 사용해서 Archimedean코푸라들과 그에 해당하는 확률밀도함수들을 계산할 수 있다. Archimedean코푸라를 계산하기 위한 MATLAB함수 couplacdf의 사용법은 각각 다음과 같다.

```
YA = copulacdf('family',U,alpha)
```

여기서 n개의 d차원 Archimedean코푸라 YA를 계산하기 위한 입력변수 family에는 Clayton, Frank 또는 Gumbel을 지정할 수 있고, 입력변수 U는 $n \times d$행렬이고, 입력변수 alpha는 각 코푸라의 모수이다.

이 MATLAB프로그램을 실행하면, 각 코푸라로부터 11^2개 벡터들을 계산한다. 이 벡터들의 등위곡선들이 그림 2.4.3에 그려져 있다. 그림 2.4.3의 좌측상단 그래프는 Clayton코푸라의 등위곡선들이, 우측상단 그래프에는 Frank코푸라의 등위곡선들이, 좌측하단 그래프에는 Gumbel코푸라의 등위곡선들이, 그리고 우측하단 그래프에는 Gauss코푸라의 등위곡선들이 그려져 있다. ■

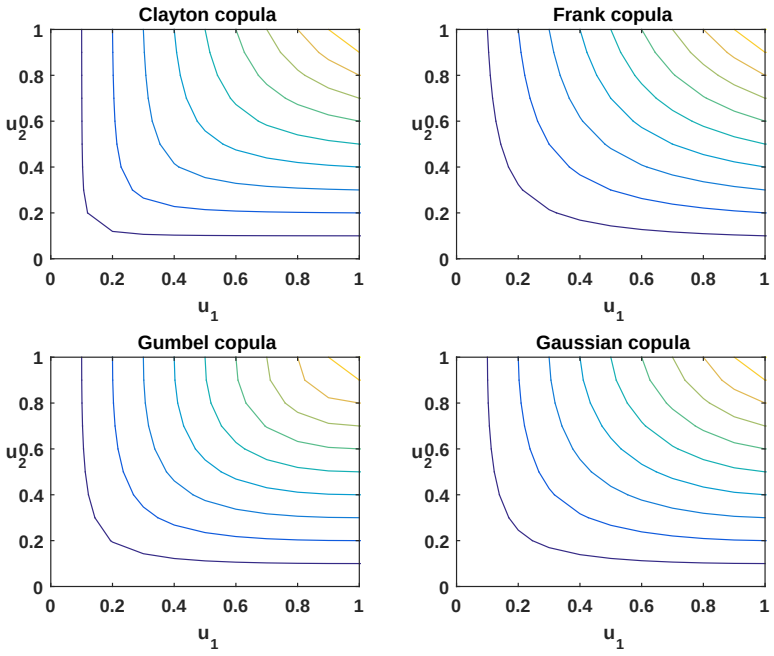


그림 2.4.3. Archimedian코푸라와 Gauss코푸라

예제 2.4.6 R을 사용해서 Archimedean코푸라들로부터 난수벡터들을 생성하는 예로서 다음 R프로그램 CopulaRVs101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------------
# Filename: CopulaRVs101R.R
# RV Contours of Archimedean Copulas
# ------------------------------------------------------------------------------
# install.packages("copula")
library("copula")
myCopula.clayton <- archmCopula(family="clayton", dim=2, param=2)
myCopula.frank   <- archmCopula(family="frank", dim=2, param=7)
myCopula.gumbel  <- archmCopula(family="gumbel", dim=2, param=2)

set.seed(1)
nSim <- 100
rvC <- rcopula(myCopula.clayton,nSim)
rvF <- rcopula(myCopula.frank,nSim)
rvG <- rcopula(myCopula.gumbel,nSim)

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('CopulaRVs101R.eps')  # Start to save figure
RVdata <- data.frame(rvC,rvF,rvG)
plotC <- ggplot(RVdata1,aes(x=rvC[,1],y=rvC[,2])) +
          geom_point(shape=16,col="red",lwd=1.5)
plotF <- ggplot(RVdata1,aes(x=rvF[,1],y=rvF[,2])) +
          geom_point(shape=17,col="green",lwd=1.5)
plotG <- ggplot(RVdata1,aes(x=rvG[,1],y=rvG[,2])) +
          geom_point(shape=18,col="blue",lwd=1.5)
plotCFG <- ggplot(RVdata1) +
        geom_point(aes(x=rvC[,1],y=rvC[,2]),shape=16,col="red",lwd=1.5) +
        geom_point(aes(x=rvF[,1],y=rvF[,2]),shape=17,col="green",lwd=1.5) +
        geom_point(aes(x=rvG[,1],y=rvG[,2]),shape=18,col="blue",lwd=1.5)
pushViewport(viewport(layout=grid.layout(2,2)))
print(plotC, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plotF, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plotG, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plotCFG, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()            # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 각 코푸라로부터 100개 난수벡터들을 생성한다. 이 벡터들의 산포도들이 그림 2.4.4에 그려져 있다. 그림 2.4.4의 좌측상단 그래프는 Clayton코푸라의 난수벡터를 적색 원으로 나타낸 산포도이고, 우측상단 그래프는 Frank코푸라의 난수벡터를 녹색 삼각형으로 나타낸 산포도이고, 좌측하단 그래프는 Gumbel코푸라의 난수벡터를 청색 다이아몬드로 나타낸 산포도이고, 우측하단 그래프는 이 세 산포도들을 합한 산포도이다. ■

예제 2.4.7 Archimedean코푸라의 결합확률밀도함수를 계산하는 예로서 다음 MATLAB

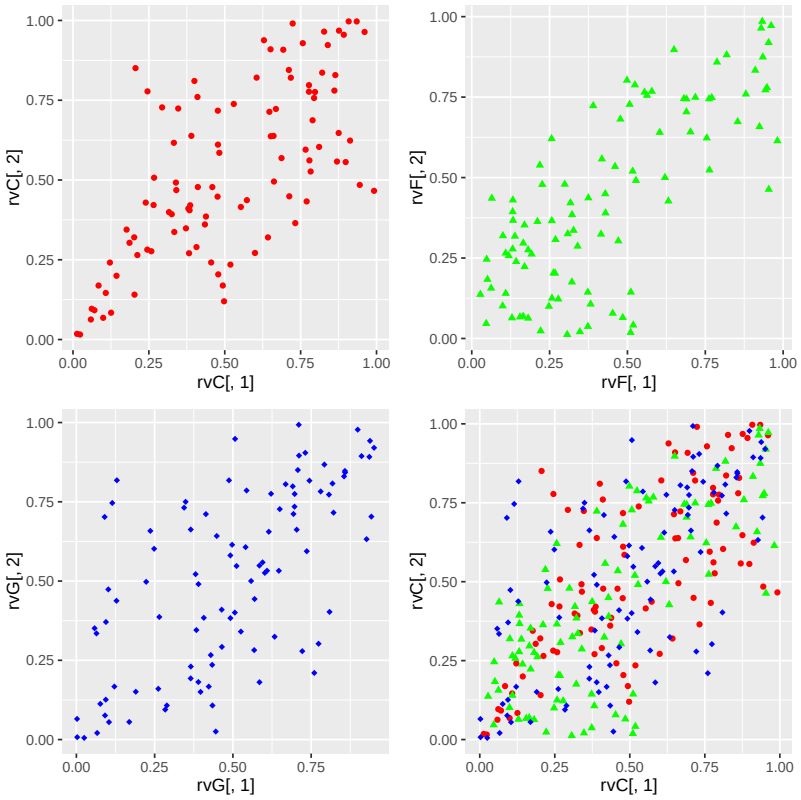


그림 2.4.4. Archimedian코푸라의 산포도

프로그램 CopulaPDFs101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: CopulaPDFs101.m
%  Copula PDF's
%  Programmed by CBS
% ------------------------------------------------------------------------
clear all, close all
nint = 11;
u = linspace(0,1,nint);
[U1,U2 ] =meshgrid(u,u);
% Clayton Copula
Fdum = copulapdf('Clayton',[U1(:) U2(:)],2);
Fc = reshape(Fdum,nint,nint);
subplot(2,2,1)
% makegray = gray(12); colormap(makegray);
surf(U1,U2,Fc)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf u_{1}','fontsize',12);
ylabel('\bf u_{2}','rotation',0,'fontsize',12);
title('\bf Clayton copula','fontsize',12)
% Frank Copula
Fdum = copulapdf('Frank',[U1(:) U2(:)],2);
Fc = reshape(Fdum,nint,nint);
subplot(2,2,2)
% makegray = gray(12); colormap(makegray);
surf(U1,U2,Fc)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf u_{1}','fontsize',12);
```

```
ylabel('\bf u_{2}','rotation',0,'fontsize',12);
title('\bf Frank copula','fontsize',12)
% Gumbel Copula
Fdum = copulapdf('Gumbel',[U1(:) U2(:)],2);
Fc = reshape(Fdum,nint,nint);
subplot(2,2,3)
% makegray = gray(12); colormap(makegray);
surf(U1,U2,Fc)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf u_{1}','fontsize',12);
ylabel('\bf u_{2}','rotation',0,'fontsize',12);
title('\bf Gumbel copula','fontsize',12)
% Gaussian Copula
Fdum = copulapdf('Gaussian',[U1(:) U2(:)],0.5);
Fc = reshape(Fdum,nint,nint);
subplot(2,2,4)
% makegray = gray(12); colormap(makegray);
surf(U1,U2,Fc)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf u_{1}','fontsize',12);
ylabel('\bf u_{2}','rotation',0,'fontsize',12);
title('\bf Gaussian copula','fontsize',12)
saveas(gcf,'CopulaPDFs101','epsc')
save('CopulaPDFs101','Fdum','Fc')
% End of program
% --------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 각 코푸라의 결합확률밀도함수로부터 11^2 개 벡터들을 계산한다. 이 벡터들의 3D곡선들이 그림 2.4.5에 그려져 있다. 그림 2.4.5의 좌측상단에는 Clayton코푸라의 결합확률밀도함수가, 우측상단에는 Frank코푸라의 결합확률밀도함수가, 좌측하단에는 Gumbel코푸라의 결합확률밀도함수, 그리고 우측하단에는 Gauss코푸라의 결합확률밀도함수가 그려져 있다. ■

정리 2.4.2에서 알 수 있듯이, 결합확률분포를 알고 있는 것과 주변확률분포들과 더불어 코푸라를 알고 있는 것은 동치이다. 따라서, 코푸라와 주변확률분포함수들을 알고 있다면, 다음 알고리즘을 사용해서 결합확률분포를 따르는 점열을 발생시킬 수 있다.

알고리즘 2.4.1: 코푸라를 사용한 난수벡터 생성

확률벡터 $\boldsymbol{y} = [y^{(1)}, y^{(2)}, \cdots, y^{(k)}]^t$ 의 결합확률분포를 따르는 점열, 즉 난수벡터열 $\left\{ \boldsymbol{y}_j = [y_j^{(1)}, y_j^{(2)}, \cdots, y_j^{(k)}]^t \,\middle|\, j = 1, 2, \cdots, N \right\}$ 를 다음과 같이 구할 수 있다.

(1단계) 초입방체 $(0,1)^k$ 에서 일양확률분포하는 난수벡터열 $\left\{ \boldsymbol{u}_j = [u_j^{(1)}, u_j^{(2)}, \cdots, u_j^{(k)}]^t \,\middle|\, j = 1, 2, \cdots, N \right\}$ 을 생성한다.

(2단계) 알고리즘 2.2.1과 조건부확률분포를 사용해서, 주변확률분포는 일양확률분포이

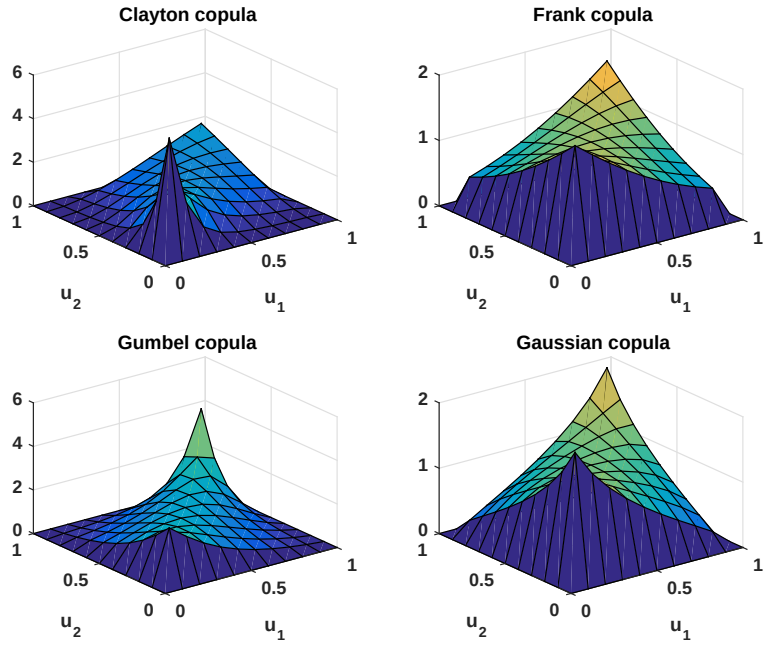


그림 2.4.5. Archimedian코푸라와 Gauss코푸라의 결합확률밀도함수들

지만 초입방체 $(0,1)^k$에서 일양확률분포하지 않는 난수벡터열 $\left\{ \boldsymbol{x}_j = [x_j^{(1)}, x_j^{(2)}, \cdots, x_j^{(k)}]^t \;\middle|\; j = 1, 2, \cdots, N \right\}$을 발생시킨다.

(3단계) Sklar정리를 이용해서, 주변확률분포들 $F^{(1)}(x_1), F^{(2)}(x_2), \cdots, F^{(k)}(x_k)$와 난수벡터열 $\left\{ \boldsymbol{x} = [x_j^{(1)}, x_j^{(2}, \cdots, x_j^{(k)}]^t \right\}$로부터 원하는 결합확률분포를 따르는 난수벡터열 $\left\{ \boldsymbol{y}_j = [y_j^{(1)}, y_j^{(2)}, \cdots, y_j^{(k)}]^t \;\middle|\; j = 1, 2, \cdots, N \right\}$을 발생시킨다.

예제 2.4.8 MATLAB을 사용해서 코푸라들로부터 난수벡터들을 생성하는 계산하는 예로서 다음 MATLAB프로그램 CopulaRandomNumber101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: CopulaRandomNumber101.m
%  Gaussian Copula Random Numbers
%  Programmed by CBS
% ------------------------------------------------------------------------
clear all, close all
tau = -0.7
rho = copulaparam('Gaussian',tau)
tauu = copulastat('Gaussian',rho)
% Random vector generation
rand('twister',5489)
nRV = 1000;
uRV = copularnd('Gaussian',rho,nRV);          % Joint Uniform RV
u_sampleCorrP = corr(uRV)
u_sampleCorrK = corr(uRV,'type','kendall')
bRV = betainv(uRV,2,3);
b_sampleCorrK = corr(bRV,'type','kendall')
```

```
% Plotting
plot(bRV(:,1),bRV(:,2),'r.','linewidth',1.3)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf b_{1}','fontsize',12);
ylabel('\bf b_{2}','rotation',0,'fontsize',12);
axis([-0.05 1.05 -0.05 1.05 ])
axis square
saveas(gcf,'CopulaRandomNumber101','epsc')
save('CopulaRandomNumber101','bRV')
% End of program
% ------------------------------------------------------------------------
```

MATLAB함수 copulaparam은 Kendall상관계수 τ로부터 선형상관계수, 즉 Pearson상관계수 ρ를 계산하는 것이다. 이 함수의 사용법은 다음과 같다.

```
rho = copulaparam('Gaussian',tau)
rho = copulaparam('t',tau,df)
rho = copulaparam('family',tau)
```

이 MATLAB프로그램을 실행하면, Kendall상관계수 tau가 -0.7인 경우에 Gauss코푸라를 사용해서 계산한 Pearson상관계수 rho는 -0.8910임을 알 수 있다.

MATLAB함수 copulastat은 선형상관계수, 즉 Pearson상관계수 ρ로부터 Kendall상관계수 τ를 계산하는 것이다. 이 함수의 사용법은 다음과 같다.

```
tau = copulastat('Gaussian',rho)
tau = copulastat('t',rho,df)
tau = copulastat('family',rho)
```

이 MATLAB프로그램을 실행하면, Pearson상관계수 rho가 -0.8910인 경우에 Kendall상관계수 tau가 -0.7임을 알 수 있다.

MATLAB함수 copularnd은 코푸라난수벡터를 생성하기 위한 것이다. 이 함수의 사용법은 다음과 같다.

```
RV = copularnd('Gaussian',rho,N)
RV = copularnd('t',rho,df,N)
RV = copularnd('family',alpha,N)
```

여기서 N은 발생시키고자 하는 난수벡터들의 개수이다. 이 MATLAB프로그램을 실행하면, Pearson상관계수 rho가 -0.8910인 경우에 Kendall상관계수 tauu가 임을 알 수 있다.

이 MATLAB프로그램을 실행하면, Pearson상관계수 rho가 -0.8910인 2차원 Gauss코푸라로부터 1000개의 난수벡터들을 생성한다. 이 코푸라난수벡터들의 Pearson상관계수는 -0.8742이고 Kendall상관계수는 -0.6904이다. 이 코푸라난수벡터들로부터 $Beta(2,3)$

확률분포에서 생성되는 베타난수벡터들을 만들기 위해서 MATLAB함수 betainv를 사용하였다. 이렇게 생성된 베타난수벡터들의 Pearson상관계수는 -0.8728이고 Kendall상관계수는 -0.6904이다. 여기서 유의할 점은 코푸라난수벡터들의 Kendall상관계수와 베타난수벡터들의 Kendall상관계수가 동일하다는 것이다.

이 MATLAB프로그램을 실행하면, 이 베타난수벡터들의 산포도가 그려진다. 이 산포도가 그림 2.4.6에 수록되어 있다. ■

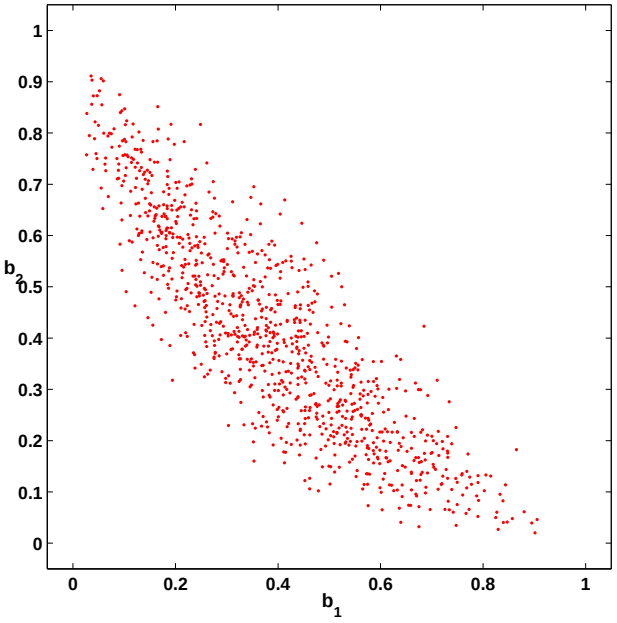


그림 2.4.6. 코푸라로 생성한 베타난수벡터

예제 2.4.9 Archimedean코푸라를 이용해서 주변확률밀도함수들을 결합해서, 결합확률밀도함수를 만드는 예로서 다음 R프로그램 CopulaMVDs101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------------
# Filename: CopulaMVDs101R.R
# MVD Contours of Archimedean Copulas with Gaussian marginals
# ------------------------------------------------------------------------------
# install.packages("copula")
library("copula")
myCopula.clayton <- archmCopula(family="clayton", dim=2, param=2)
myCopula.frank   <- archmCopula(family="frank", dim=2, param=7)
myCopula.gumbel  <- archmCopula(family="gumbel", dim=2, param=2)

# Multivariate distribution w/ given margins
myMVDc <- mvdc(copula=myCopula.clayton,
               margins=c("norm", "norm"),
               paramMargins= list(list(mean = 0, sd = 1),
                                  list(mean = 0, sd = 1)))
myMVDf <- mvdc(copula=myCopula.frank,
              margins=c("norm", "norm"),
              paramMargins= list(list(mean = 0, sd = 1),
```

```
                                  list(mean = 0, sd = 1)))
myMVDg <- mvdc( copula=myCopula.gumbel,
                margins=c("norm", "norm"),
                paramMargins= list(list(mean = 0, sd = 1),
                                   list(mean = 0, sd = 1)))
setEPS()
plot.new()
postscript('CopulaMVDs101R.eps')  # Start to save figure
par(mfrow=c(1,3), mar=c(2,2,1,1), oma=c(1,1,0,0), mgp=c(2,1,0))
contour(myMVDc, dMvdc, xlim=c(-3,3), ylim=c(-3,3), col="red")
contour(myMVDf, dMvdc, xlim=c(-3,3), ylim=c(-3,3), col="green")
contour(myMVDg, dMvdc, xlim=c(-3,3), ylim=c(-3,3), col="blue")
dev.off()            # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 표준정규확률밀도함수들인 주변확률밀도함수들 두 개에 Archimedean코푸라를 적용해서 결합밀도함수를 만든다. 이 결합밀도함수들의 등고선도가 그림 2.4.7에 그려져 있다. 그림 2.4.7의 좌측 그래프는 Clayton코푸라에 의한 결합확률밀도함수의 등고선도이고, 가운데 그래프는 Frank코푸라에 의한 결합확률밀도함수의 등고선도이고, 우측 그래프는 Gumbel코푸라에 의한 결합확률밀도함수의 등고선도이다. ■

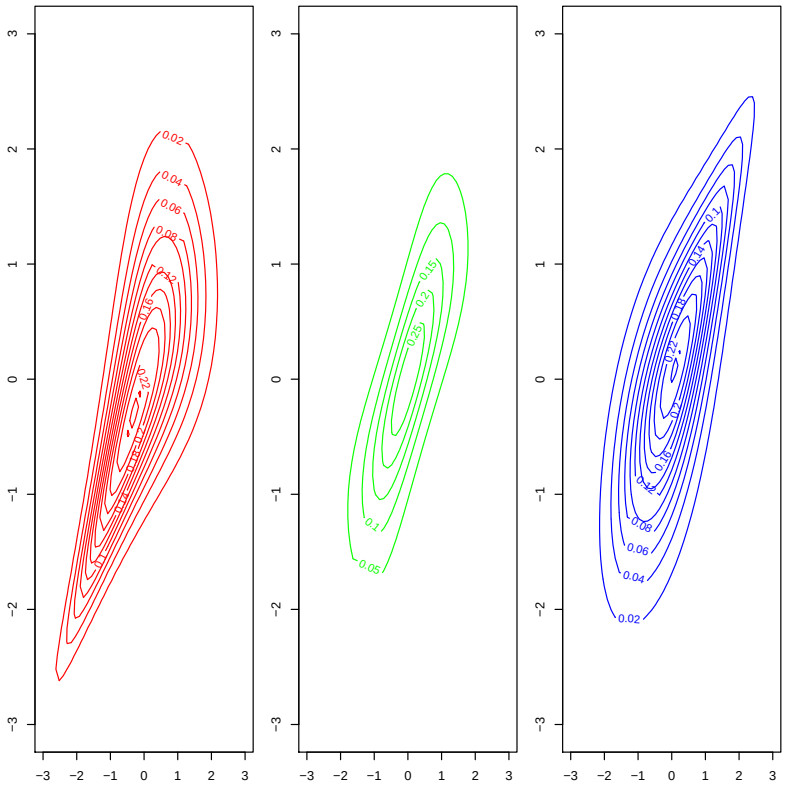


그림 2.4.7. Archimedian코푸라의 산포도

코푸라는 재무론, 도시공학, 의학, 기상학 등 넓은 범위에서 다양하게 사용되고 있다. 특히, 금융공학에서 CDO와 같은 신용파생상품의 가치평가나 위험관리에 널리 사용되고

있다. 그러나, Gauss코푸라가 2007년에 2008년에 걸친 발생한 세계경제위기의 주범 중 하나로 알려져 있고, 따라서 금융공학에서 코푸라를 사용하는데 부정적인 의견이 많다. 검은 백조 *(Black Swan)*의 저자인 N. Taleb은 가우스코푸라를 비롯한 수리재무론(quantitative finance)에 비판적인 사람이다. Taleb은 상관계수는 과거를 나타내는 것이지 미래에 발생할 일을 나타내는 것이 아니므로, 코푸라는 믿을 수 없는 것이라고 주장하며 다음과 같은 사견을 제시했다.

"Anything that relies on correlation is charlatanism."

이에 대한 자세한 내용은 Jones [21], Salmon [43]와 Donnelly & Embrechts [16]를 참조하라. 모든 분야에서 그러하듯이 도구가 잘못을 저지르는 것은 아니라, 단지 이 도구를 잘못 사용하는데서 문제가 발생하는 것이다. 어차피 인간이 상관계수를 사용해서 데이터분석을 해야 한다면, 코푸라를 사용하는 것은 피할 수 없는 일일 것이다. 다만, 코푸라를 잘 이해하고 적절히 사용해야 할 것이다.

예제 2.4.10 확률함수 $P_i(t) = P(\tau_i \le t)$가 시점 t의 결정적 함수인 경우에, 코푸라 C에 의해 디폴트시점의 결합확률분포함수가 다음과 같이 정해진다고 하자.

$$P(\tau_1 \le t_1, \tau_2 \le t_2, \cdots, \tau_n \le t_n) = C(P_1(t_1), P_2(t_2), \cdots, P_n(t_n)) \tag{1}$$

식 (1)을 사용해서 각 참가기업의 디폴트시점을 모의실험할 수 있다. 이 방법에서는 선형상관이 아닌 순위상관을 이용한다. 그러나, 이 방법에서는 어떠한 코푸라를 사용하는가에 따라 결과가 크게 달라질 수 있다.

디폴트시점을 다음과 같이 표현할 수 있다.

$$\tau_i = \inf\left\{ t \;\middle|\; \exp\left(-\int_0^t \lambda_i(s) ds \right) \le u_i \right\} \tag{2}$$

여기서 $\lambda_i(\cdot)$는 해저드율함수(hazard rate function)이고, u_i는 지지대가 $[0,1]$인 일양확률변수이다. 식 (2)를 바탕으로 u_i의 순위상관을 코푸라로 표현함으로써, 디폴트시점을 분석하는 접근법도 있다. ■

제 3 장

몬테카를로법

이 장에서는 몬테카를로법(Monte Carlo method)에 대해 간략히 소개하고자 한다. 좀 더 자세한 내용은 Glasserman [17], 최병선 [2, 4장], Korn & Korn & Kroisandt [25], Kroese & Taimre & Botev [26] 그리고 Brandimarte [10]를 참조하라.

제3.1절 몬테카를로법이란 무엇인가?

시뮬레이션(simulation)이란 실제 현상 또는 시스템을 모방한 것으로서 모의실험이라고 번역하기도 한다. 현실세계에서 우리가 풀어야하는 많은 문제들은 그 구조가 복잡해서, 그 속에서 일어나는 현상을 정확하게 파악하기 어려운 경우가 많다. 또한, 실제 실험을 실행하는데 많은 비용이 들거나, 경제현상과 같이 실험을 하는 것이 불가능한 경우도 많다. 그러한 경우, 실제 시스템을 상대적으로 단순하고 비용이 적게 드는 모형으로 정형화해서, 원래 시스템의 성능이나 움직임을 파악할 필요가 있다. 컴퓨터를 사용한 비행기조종사를 훈련시키기 위한 플라잇시뮬레이터(flight simulator)나 시뮬레이션게임 등은 그 좋은 예이다. 더구나, 시스템의 움직임을 수식으로 나타낼 수 없거나, 설사 수식으로 표현할 수 있어도 그 해를 해석적으로 구하는 것이 어려운 경우에, 시뮬레이션은 근사적인 해를 얻는 강력한 수단이다. 시뮬레이션에는 컴퓨터가 필수적이다. 즉, 컴퓨터 상에 가상현실의 세계를 만들어, 그곳에서 실험을 하는 것이다. 따라서, 세상이 복잡해지고 컴퓨터가 발전함에 따라 시뮬레이션의 가능성과 중요성이 점점 커져가는 것은 당연하다.

우리가 다루는 문제에는 체계적 움직임과 더불어 우연성을 갖는 확률적(stochastic) 움직임이 동반하는 것이 보통이다. 이러한 확률적 현상을 시뮬레이션하기 위해서는 샘플(random number)를 이용한다. 샘플를 이용한 시뮬레이션을 몬테카를로법 또는 몬테카를로실험(Monte

Carlo experiment)이라 부른다. 어떤 수리적 문제에서 닫힌해(closed-form solution)를 구할 수 없거나 또는 닫힌해가 너무 복잡한 경우에, 몬테카를로법을 적용할 수 있다.

몬테카를로법의 가장 중요한 아이디어는 확률변수 x의 확률분포로부터 서로 독립인 관찰값들 $x_1, x_2, \cdots, x_n$을 발생시켜 이들의 표본평균 $\bar{x} = \frac{1}{n}\sum_{j=1}^{n} x(j)$를 기대값 $E(x)$의 근사값으로 사용하는 것이다. 즉, 몬테카를로법에서는 먼저 분석하고자 하는 현상을 확률변수를 포함한 모형으로 모형화하고, 샘플를 생성해서 이 모형이 내포한 확률적 움직임을 나타내고, 그 결과를 대수법칙(law of large numbers: LLN)과 중심극한정리(central limit theorem: CLT)로 분석해서, 현상을 설명하거나 미래를 예측한다. 기대값은 수열의 합이나 정적분이므로, 몬테카를로법은 수열의 합이나 정적분의 근사값을 구하는 방법이라고 할 수 있다. 백문이불여일견(百聞不如一見)이라! 우선 예제를 하나 살펴보자.

예제 3.1.1 확률밀도함수가 $f(x)$인 확률분포로부터 서로 독립인 표본들 $\{x(j) \mid j = 1, 2, \cdots\}$를 추출한다고 하자. 대수법칙에 의해서, 다음 식이 성립한다.

$$\lim_{n\to\infty} \frac{1}{n}\sum_{j=1}^{n} g\left(x(j)\right) = E\left(g(x)\right) \tag{1}$$

따라서, 아주 큰 n에 대해서 다음 근사식이 성립한다.

$$\int g(x)f(x)dx \approx \frac{1}{n}\sum_{j=1}^{n} g\left(x(j)\right) \tag{2}$$

확률변수 x가 모수가 $\lambda = 1$인 지수분포를 따를 때, 몬테카를로법을 사용해서 기대값 $E\left(\sqrt{x}\right)$를 계산하기 위해서, 다음 MATLAB프로그램 MonteCarloIntegration101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
% Filename: MonteCarloIntegration101.m
% Monte Carlo Integration 1
% Programmed by CBS
% ------------------------------------------------------------------------
% Numerical integration
integrand = 'sqrt(x).*exp(-x)';
myfun = inline(integrand);
Nint = quadl(myfun,0,100)
% Generate exponential random numbers with lambda = 1
rand('twister',5489);   % Use twister or state for seed.
x = exprnd(1,1,6000);
xsqrt = sqrt(x);
% Monte Carlo integration
nn = 1:1:30;
Nsamples = nn*100;
for ndum = 1:1:30
```

```
    xsq = xsqrt(1:Nsamples(ndum));
    MCint(ndum) = mean(xsq)
end
% Ploting
plot(Nsamples,MCint,'k',[0,Nsamples(end)], ...
     [Nint,Nint],'r--','LineWidth',2.5)
set(gca,'fontsize',11,'fontweigh','bold')
legend('Monte Carlo Integration','Numerical Integration',4)
axis([0,Nsamples(end), 0.825 0.925 ])
xlabel('\it Number of Samples','Fontsize',12','Fontweigh','bold');
ylabel('\it Integration','Fontsize',12','Fontweigh','bold');
hold off
saveas(gcf,'MonteCarloIntegration101','epsc')
save('MonteCarloIntegration101','MCint','Nint')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램 MonteCarloIntegration101.m을 실행하면, 다음 정적분을 몬테카를로법과 Gauss구적법으로 계산한다.

$$I = \int_0^\infty \sqrt{x} e^{-x} dx \tag{3}$$

Gauss구적법에 의한 적분값은 0.8862이다. 또한, 표본들의 개수를 100에서 100씩 증가시켜서 최대 6000개 까지 계산한 몬테카를로적분값들이 그림 3.1.1에 그려져 있다. 그림 3.1.1에서 알 수 있듯이, 표본들의 개수 n이 충분히 커지면, 몬테카를로적분값은 참값에 가까워진다.

뒤에서 자세히 다루겠지만, 식 (1)이 성립하기 위해서 반드시 표본들 $\{x(j)\}$가 서로 독립일 필요는 없다. 만약 $f(x)$를 불변분포의 확률밀도함수로 하는 Markov체인를 만들면, Markov체인의 에르고딕성에 의해서 식 (1)이 성립함을 알 수 있다. 즉, 표본들이 독립이 아닌 MCMC에서도 근사식 (2)를 적용할 수 있다. 그러나, 이 경우에는 표본들 사이에 존재하는 상호종속성 때문에, 식 (2)에 의해 계산된 추정값의 분산에 유의해야 한다. ■

예제 3.1.2 R언어를 사용해서 예제 3.1.1의 문제를 풀기 위해, 다음 R프로그램 MonteCarloIntegration101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: MonteCaroIntegration101R.R
# Monte Carlo Integration
# Programmed by CBS
# ------------------------------------------------------------------------
# True Definite Integral
myftn <- function(x) x^0.5*exp(-x) # Integrand it over (0,Inf)
( DefInt1 <- integrate(myftn, lower=0, upper=Inf) )

```

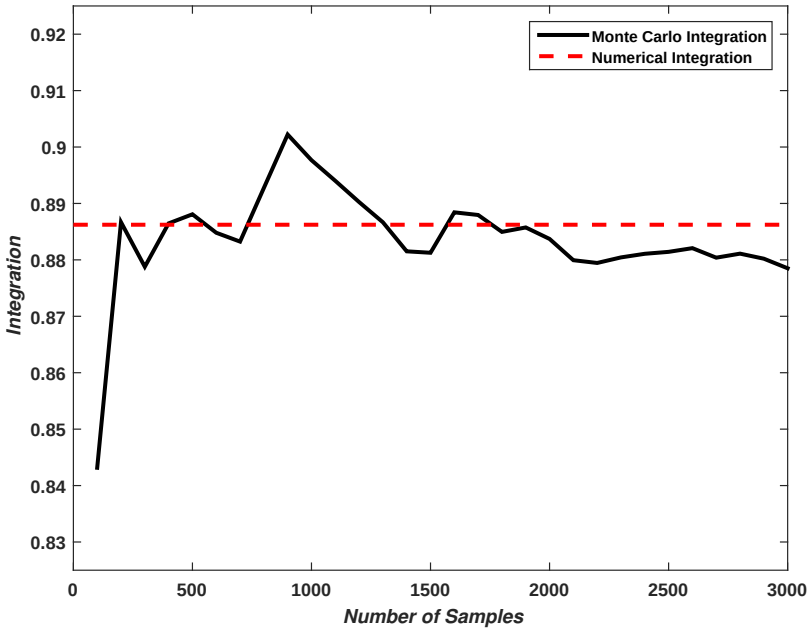


그림 3.1.1. 몬테카를로적분과 MATLAB

```
# Generating Random Numbers
set.seed(11)
nSim <- 6000

x <- rexp(nSim,1)
xsqrt <- sqrt(x)
MCint <- matrix(0,30,1)
# Monte Carlo Integration
nn <- c(1:1:30)
Nsamples <- nn*100
for (ndum in 1:1:30){
     xsq <- xsqrt[1:Nsamples[ndum]]
     MCint[ndum] <- mean(xsq)
}

# Plotting
# install.packages("ggplot2")
library(ggplot2)
setEPS()
plot.new()
postscript('MonteCaroIntegration101R.eps')  # Start to save figure
RVdata1 <- data.frame(Nsamples,MCint)
ggplot(RVdata1) +
       geom_line(aes(x=Nsamples,y=MCint),col="red",lwd=1.2) +
       geom_point(aes(x=Nsamples,y=MCint),col="black",lwd=1.5) +
       geom_hline(yintercept=0.8862,col="dark blue",lwd=1) +
       ylim(0.825,0.925) +
       xlab("Number of Samples") + ylab("Intgration")
dev.off()            # End to save figure
# ------------------------------------------------------------------------
```

이 R프로그램 MonteCarloIntegration101R.R을 실행하면, 다음 정적분을 계산한다.

$$I = \int_0^\infty \sqrt{x} e^{-x} dx \tag{3}$$

이 적분값은 0.8862이다. 또한, 표본들의 개수를 100에서 100씩 증가시켜서 최대 6000개까지 계산한 몬테카를로적분값들이 그림 3.1.2에 그려져 있다. 그림 3.1.2에서 알 수 있듯이, 표본들의 개수 n이 충분히 커지면, 몬테카를로적분값은 참값에 가까워진다. ■

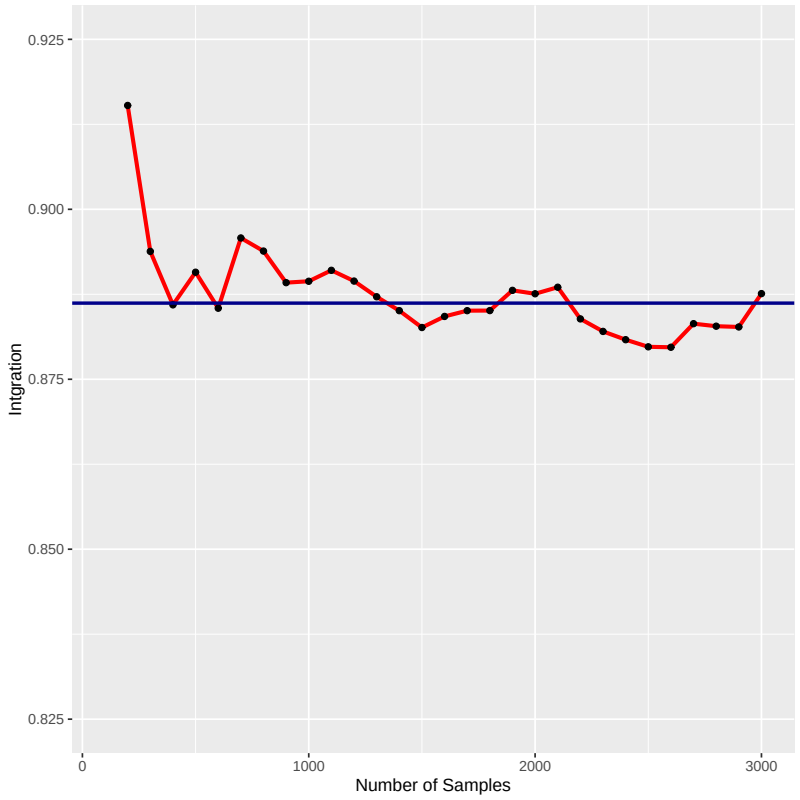


그림 3.1.2. 몬테카를로적분과 R

몬테카를로법이 개발되기 이전에는 이미 해가 알려진 결정적(deterministic) 문제를 시뮬레이션할 때, 이 시뮬레이션에서 발생하는 불확실성을 추정하기 위해서 통계적 샘플링이 사용되었다. 역으로, 몬테카를로법에서는 결정적 문제를 풀기 위해서 확률적 접근법을 사용한다. 예를 들어, Georges Louis Leclerc, Comte de Buffon (1707-1788)이 제시한 Buffon의 바늘문제(Buffon's needle problem)를 바탕으로 원주율 π를 계산하는데 사용되었다. 먼저 이 문제를 살펴보자.

예제 3.1.3 Buffon의 바늘문제는 한변의 길이가 d인 정사각형 타일이 무한히 연결되어 있는

무한평면에 길이가 $l(< d)$ 인 바늘을 떨어트렸을 때 바늘이 타일의 변과 만날 확률을 구하는 것이다. 이 Buffon의 바늘문제를 풀기 위해서 그림 3.1.3를 참조하라. 그림 3.1.3를 그리기 위해서 다음 MATLAB프로그램 BuffonNeedleProblem101.m을 실행하라.

```
% ------------------------------------------------------------------
% Filename: BuffonNeedleProblem101
% Buffon's Needle Problem
% Programmed by CBS
% ------------------------------------------------------------------
clear, clf
% Plotting
plot([ -0.5 1.5 ], [0 0],'b--','linewidth',1.5)
hold on
plot([ -0.5 1.5 ], [1 1],'b--','linewidth',1.5)
plot([0 0],[ -0.5 1.5 ],'b--','linewidth',1.5)
plot([1 1],[ -0.5 1.5 ],'b--','linewidth',1.5)
set(gca,'fontsize',11,'fontweigh','bold')
x0 = 0.2; y0 = 0.3;
theta1 = 0.2*pi;
x1 = x0+0.5*cos(theta1); y1 = y0 + sin(theta1);
plot([x0 x1],[ y0 y1 ],'r-','linewidth',3)
plot([x0 x0],[ y0 y1 ],'k-','linewidth',2)
plot([x0 x1],[ y1 y1 ],'k-','linewidth',2)
plot([0 0],[ y0 1 ],'k-','linewidth',2)
plot([0 x0],[ y0 y0 ],'m-.','linewidth',2)
axis( [-0.5 1.5 -0.5 1.5 ])
axis square
set(gca,'xtick',[0 1],'xticklabel',{'0' 'd'})
set(gca,'ytick',[0 1],'yticklabel',{'0' 'd'})
text(x1-0.12, y1-0.04, '\bf \theta')
text( (x1+x0)/2, (y1+y0)/2-0.02, '\bf\it l')
text( x0, (y1+y0)/2, '\bf\it h')
text(-0.1, (y0+1)/2, '\bf\it H')
text( x0-0.05, y0-0.05, '\bf\it A')
text( x0-0.07, y1+0.03, '\bf\it C')
text( x1, y1, '\bf\it B')
hold off
saveas(gcf,'BuffonNeedleProblem101','epsc')
% End of program
% ------------------------------------------------------------------
```

그림 3.1.3에서 점 A와 점 B가 길이가 l인 바늘의 양끝을 나타낸다고 하자. 점 C는 선분 AC와 선분 BC가 타일의 변들과 평행인 점이다. 선분 AB와 선분 BC가 이루는 각을 θ라 하면, 선분 AC의 길이 h는 $l\sin\theta$이다. 반직선 $\underset{AC}{\rightarrow}$가 타일의 변과 만나는 최단거리를 H라고 하면, 다음 조건이 성립하면 바늘은 타일의 변과 만난다.

$$H \le h = l\sin\theta \tag{1}$$

따라서, 바늘이 타일의 변과 만날 확률 p은 다음과 같다.

$$p = \frac{\int_0^{\pi}\int_0^{l\sin\theta} 1dH\,d\theta}{\int_0^{\pi}\int_0^{d} 1dH\,d\theta} = \frac{\int_0^{\pi} l\sin\theta d\theta}{\pi d} = \frac{2l}{\pi d} \tag{2}$$

■

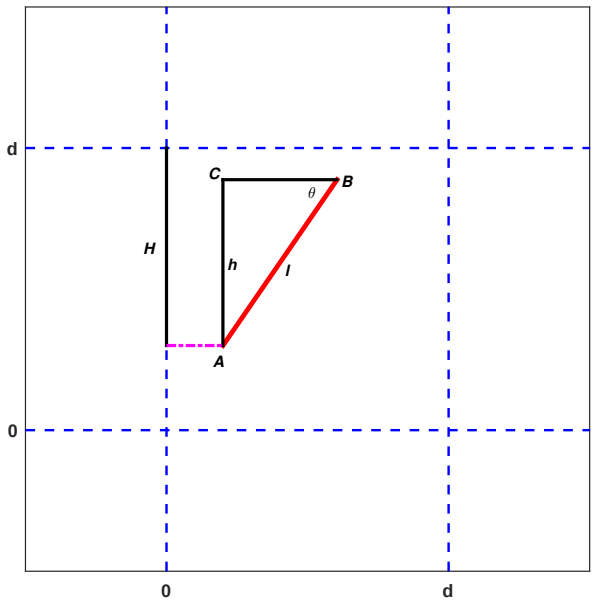


그림 3.1.3. Buffon의 바늘문제

예제 3.1.3의 결과를 사용해서, π의 근사값을 구하기로 하자. 다음 웹사이트에서 이 실험에 관한 애플릿(applet)을 제공한다.

`http://www.angelfire.com/wa/hurben/buff.html`

예제 3.1.4 예제 3.1.3에서 알 수 있듯이, Buffon의 바늘문제는 한변의 길이가 d인 정사각형 타일이 무한히 연결되어 있는 무한평면에 길이가 $l = d/2$인 바늘을 떨어트렸을 때 바늘이 타일의 변과 만날 확률은 $p = 1/\pi$이다. 따라서, 이 바늘을 N번 떨어트렸을 때 정사각형의 변과 만나는 회수가 n번이면, 다음 근사식이 성립한다.

$$\pi \approx \frac{N}{n} \tag{1}$$

식 (1)과 몬테카를로법을 사용해서 원주율 π를 계산하기 위해서, 다음 MATLAB프로그램 MCBuffonNeedle101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
% Filename: MCBuffonNeedle101.m
% Monte Carlo solution of Buffon's Needle Problem
% Programmed by CBS
% ------------------------------------------------------------------------
clear all, clf, close all
rand('twister',5489);   % Use twister or state for seed.
NN = 10000*20;
HTot = rand(1,NN);
thetaTot = pi/2*rand(1,NN);
for ii=1:1:20
    NoThrows(ii) = 10000*ii;              % Number of NoThrows
    dumHits = (HTot(1:NoThrows(ii)) <= ...
               0.5*sin(thetaTot(1:NoThrows(ii))) );
    SumHits(ii) = sum(dumHits);           % No of Hitting a border
    pihat(ii) = NoThrows(ii)/SumHits(ii); % Pr of Hitting a border
    errorr(ii) = pi - pihat(ii);
end
pihat, errorr
% Plotting
subplot(2,1,1)
plot(NoThrows,pihat,'k-o','linewidth',1.5)
set(gca,'fontsize',11,'fontweigh','bold')
hold on
plot([NoThrows(1) NoThrows(20)],[pi pi],'r-','linewidth',1.2)
ylabel('\bf Estimate','Fontsize',11');
hold off
axis([10^4 2*10^5 3.1 3.2 ])
subplot(2,1,2)
plot(NoThrows,errorr,'r--*','linewidth',1.5)
set(gca,'fontsize',11,'fontweigh','bold')
ylabel('\bf Error','Fontsize',11');
hold on
plot([NoThrows(1) NoThrows(20)],[0 0],'k-','linewidth',1.2)
hold off
axis([10^4 2*10^5 -0.06 0.06 ])
saveas(gcf,'MCBuffonNeedle101','epsc')
save('MCBuffonNeedle101','pihat','errorr')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램 MCBuffonNeedle101.m을 실행하면, 표본들의 개수를 10000에서 10000씩 증가시켜서 계산한 π의 근사값과 오차가 그림 3.1.4 에 그려져 있다. 그림 3.1.4 의 상단 그래프에서 알 수 있듯이, 표본들의 개수 n이 충분히 커지면 몬테카를로법으로 계산한 값은 π에 가까워진다. 하단 그래프는 π와 몬테카를로값의 차이를 그린 것이다. ■

예제 3.1.5 R언어를 사용해서 예제 3.1.4의 문제를 풀기 위해, 다음 R프로그램 MCBuffonNeedle101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: MCBuffonNeedle101R.R
```

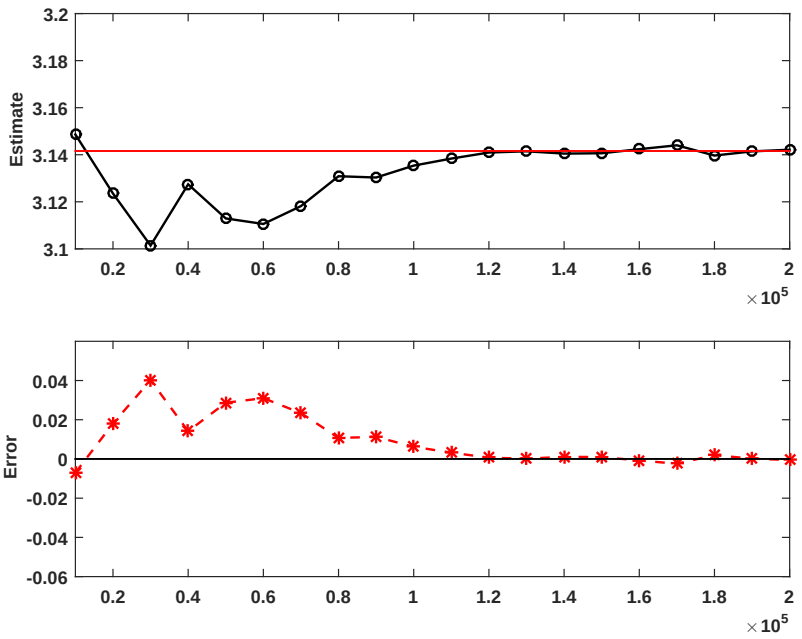


그림 3.1.4. MATLAB과 몬테카를로법을 사용한 원주율 계산

```
# Monte Carlo Solution of Buffon's Needle Problem
# Programmed by CBS
# ------------------------------------------------------------------------------
# Generating Random Numbers
set.seed(11)
NN <- 10000*20
HTot <- runif(NN, min = 0, max = 1)
thetaTot <- pi/2*runif(NN)

# Monte Carlo Integration
NoThrows <- rep(0,20)
Pihat    <- rep(0,20)
errorr   <- rep(0,20)
SumHits  <- rep(0,20)

for (ii in 1:1:20){
     NoThrows[ii] <- 10000*ii
     dumHits <- ( HTot[1:NoThrows[ii]] <=
                  0.5*sin(thetaTot[1:NoThrows[ii]]) )
     SumHits[ii] <- sum(dumHits)
     Pihat[ii] <- NoThrows[ii]/SumHits[ii]
     errorr[ii] <- pi - Pihat[ii]
}

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('MCBuffonNeedle101R.eps')  # Start to save figure
nSim <- 10000*c(1:1:20)
RVdata <- data.frame(nSim,Pihat,errorr)
plotMC <- ggplot(RVdata,aes(x=nSim,y=Pihat)) +
          geom_point(shape=16,col="black",lwd=1.7)+
```

```
39            geom_line(col="red") +
40            geom_hline(yintercept=pi,col="dark blue",lwd=1) +
41            ylim(3.1,3.2) +
42            xlab("Number of Samples") + ylab("Estimate")
43  plotEr <- ggplot(RVdata,aes(x=nSim,y=errorr)) +
44            geom_point(shape=16,col="black",lwd=1.7) +
45            geom_line(col="green") +
46            geom_hline(yintercept=0.0,col="dark blue",lwd=1) +
47            ylim(-0.06,0.06) +
48            xlab("Number of Samples") + ylab("Error")
49  pushViewport(viewport(layout=grid.layout(2,1)))
50  print(plotMC, vp=viewport(layout.pos.row=1, layout.pos.col=1))
51  print(plotEr, vp=viewport(layout.pos.row=2, layout.pos.col=1))
52  dev.off()             # End to save figure
53  # ------------------------------------------------------------------------------
```

이 R프로그램 MCBuffonNeedle101R.m을 실행하면, 표본들의 개수를 10000에서 10000씩 증가시켜서 계산한 π의 근사값과 오차가 그림 3.1.5 에 그려져 있다. 그림 3.1.5 의 상단 그래프에서 알 수 있듯이, 표본들의 개수 n이 충분히 커지면 몬테카를로법으로 계산한 값은 π에 가까워진다. 하단 그래프는 π와 몬테카를로값의 차이를 그린 것이다. ■

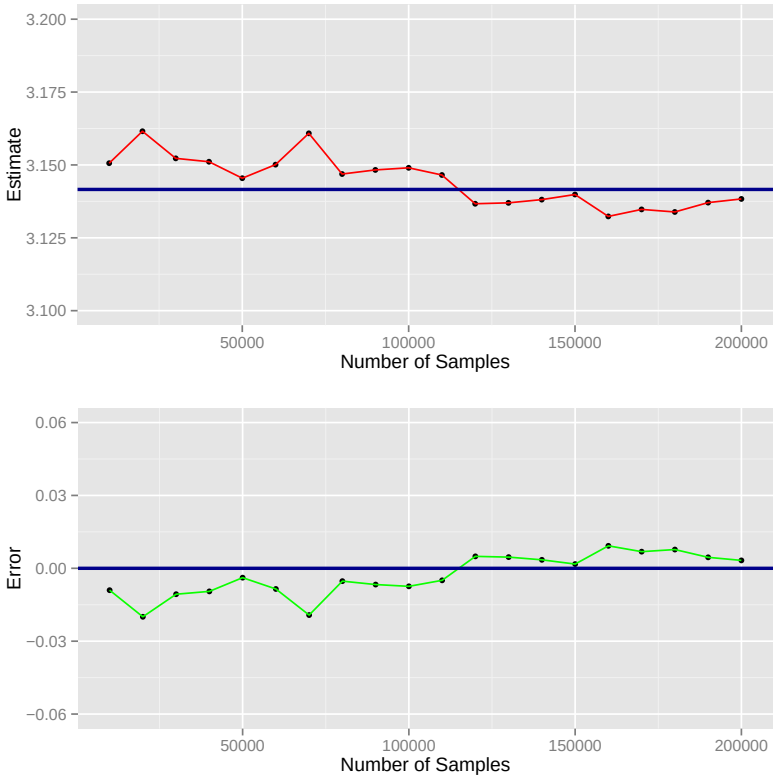


그림 3.1.5. R과 몬테카를로법을 사용한 원주율 계산

Fermi가 1930년대에 중성자확산(neutron diffusion)에 관한 연구를 하면서 처음으로

몬테카를로실험을 했다고 알려져 있다. 그러나, 1946년 Los Alamos연구소(Los Alamos Scientific Laboratory)에서 몬테카를로법을 사용해서 방사능방패(radiation shielding)와 중성자(neutron)가 움직이는 거리의 관계를 규명하면서, 이 방법이 체계적으로 연구되고 널리 사용되기 시작하였다. 이 방법을 제시한 것은 Ulam이다. 그는 조합이론(combinatorics)을 사용해서 카드게임의 하나인 캔필드솔리테어게임(Canfield solitaire game)에서 이길 수 있는 확률을 계산하던 중, 복잡한 조합이론을 적용하는 것보다는 100번 게임을 하고 이 중에서 이기는 회수 n을 100으로 나눈 $n/100$을 근사확률로 채택하는 것이 훨씬 더 실용적이라는 것을 알았다. 이러한 과정에서, 이 방법을 중성자확산문제(problem of neutron diffusion)와 같은 수리물리문제나 미분방정식을 푸는데 적용할 수 있다는 것을 깨달았다고 한다. 이 해에 Ulam은 von Neumann에게 이 방법을 설명하였고, 이들은 이 방법을 적용하는 연구를 시작했다. 이 연구의 비밀을 유지하기 위해 작전명(code name)으로 도박장으로 유명한 '몬테카를로'를 사용하였다. 몬테카를로법은 맨하탄프로젝트(Manhattan project)에서 사용된 주된 시뮬레이션기법이었지만, 당시에는 컴퓨터가 발전하지 않아서 큰 역할을 하지는 못했다고 한다. 맨하탄프로젝트는 1942년에서 1946년까지 진행된 미국이 주도하고 영국과 캐나다가 공동으로 참여한 핵폭탄 개발프로그램이다. Metropolis & Ulam (1949)이 몬테카를로법에 관한 첫 번째 논문을 발표하였으며, 1950년대에 Los Alamos연구소에서 수소폭탄의 개발에 몬테카를로법이 사용되었고, 그 후 여러 학문분야에서 널리 사용되고 있다. 심지어는 초등수학에서 나타나는 복잡한 문제도 몬테카를로법을 사용해서 풀 수가 있다.

예제 3.1.6 미국 고등학교 수학경시대회의 예선이라고 할 수 있는 AMC12에서 출제된 다음 문제(2010년 AMC12B Problem 18)를 몬테카를로법을 사용해서 풀어보자.

> (문제) 한 번에 1m미터 씩 점프하는 황소개구리가 반경이 1m인 원의 중심에 있다. 이 황소개구리가 연속해서 세 번 점프해서 착지한 곳이 이 원의 원주 상이나 내부에 있을 확률을 구하라.

황소개구리가 첫 번째 점프를 나타내는 벡터를 $\boldsymbol{x}_1$, 두 번째 점프를 나타내는 벡터를 $\boldsymbol{x}_2$, 그리고 세 번째 점프를 나타내는 벡터를 $\boldsymbol{x}_3$라 하자. 이 문제는 조건 하에서 식 $||\boldsymbol{x}_1||+||\boldsymbol{x}_2||+||\boldsymbol{x}_3|| \leq 1$이 성립할 확률을 구하는 것이다. 이 문제를 풀기 위해서는, 각 벡터를 다음과 같이 극좌표로 나타내는 것이 편리할 것이다.

$$\boldsymbol{x(i)} = [\cos\theta(i), \sin\theta(i)]\,, \quad (i = 1, 2, 3) \tag{1}$$

즉, 우리의 문제는 다음 식을 만족하는 확률을 구하는 것이다.

$$\sqrt{\left[\sum_{i=1}^{3}\cos\theta(i)\right]^2+\left[\sum_{i=1}^{3}\sin\theta(i)\right]^2}\le 1 \tag{2}$$

몬테카를로법을 적용해서 이 확률을 구하기 위해서는 먼저 아주 큰 n에 대해서 구간 $[0,2\pi]$에서 일양샘플들(uniform random numbers) $\{\theta_{1,j}\mid j=1,2,\cdots,n\}$, $\{\theta_{2,j}\mid j=1,2,\cdots,n\}$와 $\{\theta_{3,j}\mid j=1,2,\cdots,n\}$를 생성한다. 다음 단계로, 식 (2)를 만족하는 $[\theta_{1,j},\theta_{2,j},\theta_{3,j}]$의 개수를 n으로 나눈 것이 우리가 구하고자 하는 확률의 근사값이다.

이 문제를 기하학적으로 살펴보기로 하자. 다음 식들이 성립함을 쉽게 알 수 있다.

$$\begin{aligned}||\boldsymbol{x}_1+\boldsymbol{x}_2+\boldsymbol{x}_3||^2&=\left[\sum_{i=1}^{3}\cos\theta(i)\right]^2+\left[\sum_{i=1}^{3}\sin\theta(i)\right]^2\\&=3+2\left[\cos(\theta_1-\theta_2)+\cos(\theta_2-\theta_3)+\cos(\theta_3-\theta_1)\right]\end{aligned} \tag{3}$$

따라서, 우리의 문제는 다음 식을 만족하는 확률을 구하는 것이다.

$$\cos(\theta_1-\theta_2)+\cos(\theta_2-\theta_3)+\cos([\theta_3-\theta_2]-[\theta_2-\theta_1])\le -1 \tag{4}$$

몬테카를로법을 사용해서 식 (2)를 만족하는 확률을 계산하기 위해서, 다음 MATLAB 프로그램 MonteCarloProbability101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
% Filename: MonteCarloProbability101.m
% Monte Carlo Solution of 2010 AMC12B Problem 18
% Programmed by CBS
% ------------------------------------------------------------------------
clear all, clf
% Monte Carlo Simulation
rand('twister',5489);   % Use twister or state for seed.
n = 100000;  % Number of random samples
theta1 = 2*pi*rand(n,1);
theta2 = 2*pi*rand(n,1);
theta3 = 2*pi*rand(n,1);
x = cos(theta1) + cos(theta2) + cos(theta3);
y = sin(theta1) + sin(theta2) + sin(theta3);
r = sqrt(x.^2 + y.^2);
Inside = ( r <= 1);
prob = sum(Inside)/n
% Plotting
nn = 100;
tt21 = -2*pi:4*pi/(nn-1):2*pi;
tt31= -2*pi:4*pi/(nn-1):2*pi;
plot(pi,pi,'r*','linewidth',1)
set(gca,'fontsize',11,'fontweigh','bold')
```

```
hold on
for n21 = 1:1:nn
    for n31 = 1:1:nn
        if cos(tt21(n21))+cos(tt31(n31)) ...
                +cos(tt21(n21)+tt31(n31))+1 <= 0
           plot(tt21(n21),tt31(n31),'r*','linewidth',1)
        end
    end
end
hold off
axis( [ -2*pi 2*pi -2*pi 2*pi ])
axis square
xlabel('\bf \theta_{2} - \theta_{1}','Fontsize',12');
ylabel('\bf \theta_{3} - \theta_{1}','Fontsize',12');
saveas(gcf,'MonteCarloProbability101','epsc')
% End of program
% --------------------------------------------------------------------------
```

이 MATLAB프로그램 MonteCarloProbability101.m을 실행하면, 식 (2)를 만족하는 확률을 구한다. 표본수를 $n = 100,000$으로 했을 때, 확률이 0.2507을 출력하였다. 물론, 이 값은 실험 때마다 약간씩 달라진다. 또한, 식 (4)가 만족되는 부분을 그린 그림 3.1.6를 출력한다. 그림 3.1.6에서 알 수 있듯이, 식 (4)를 만족하는 확률은 0.250이다. ■

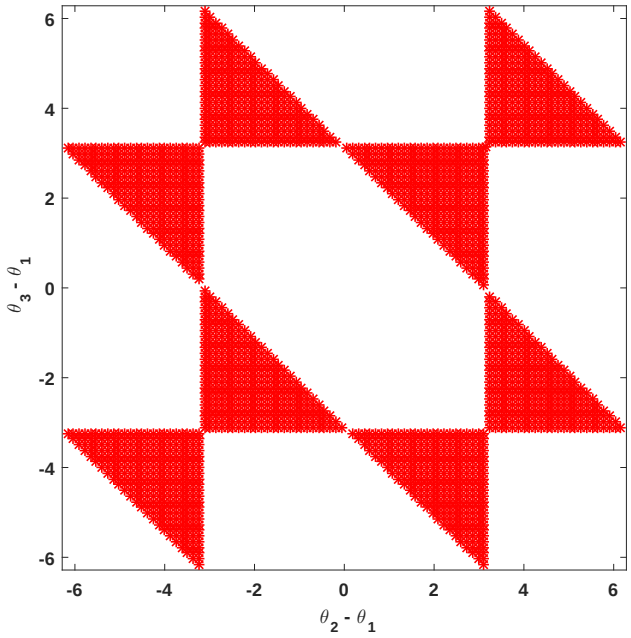


그림 3.1.6. 몬테카를로적분

예제 3.1.7 R언어를 사용해서 예제 3.1.6의 문제를 풀기 위해, 다음 R프로그램 /MCBuffonNeedle101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------------
# Filename: MonteCaroProbability101R.R
# Monte Carlo Integration
# Programmed by CBS
# ------------------------------------------------------------------------------
# Generating Random Numbers
set.seed(11)
nSim <- 100000
theta1 <- 2*pi*runif(nSim,0,1)
theta2 <- 2*pi*runif(nSim,0,1)
theta3 <- 2*pi*runif(nSim,0,1)
x <- cos(theta1) + cos(theta2) + cos(theta3)
y <- sin(theta1) + sin(theta2) + sin(theta3)
r <- sqrt(x^2 + y^2)
Inside <- (r <= 1)
prob <- sum(Inside)/nSim
prob

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages('latex2exp')
library(latex2exp)
setEPS()
plot.new()
postscript('MonteCarloProbability101R.eps')  # Start to save figure
nn <- 100
xdum <- seq(-2*pi,2*pi,length.out=nn)
tt <- expand.grid(x=xdum, y=xdum)
tt$z <- 0+( cos(tt$x) + cos(tt$y) + cos(tt$x+tt$y) +1 <= 0 )
RVdata <- data.frame(tt)
ggplot(RVdata,aes(x=tt$x, y=tt$y, fill=tt$z)) +
   geom_raster() +
   xlab(TeX('$\\theta_{2} - \\theta_{1}$')) +
   ylab(TeX('$\\theta_{3} - \\theta_{1}$'))
# (cf) xlab(TeX('$\\alpha  x^\\alpha$, where $\\alpha \\in 1\\ldots 5$'))
dev.off()            # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램 MonteCarloProbability101R.m을 실행하면, 표본수가 $n = 100,000$인 경우 확률이 0.2513로 추정된다. 물론, 이 값은 실험 때마다 약간씩 달라진다. 또한, 그림 3.1.7가 출력된다. 그림 3.1.7에서 알 수 있듯이, 식 (4)를 만족하는 확률은 0.250이다. ■

제3.2절 대수법칙과 중심극한정리

앞에서도 언급했듯이, 몬테카를로법은 대수법칙을 바탕으로 한다. 또한, 몬테카를로법에 의한 결과의 타당성을 조사하기 위해서는 중심극한정리(the central limit theorem: CLT)를 필요로 한다. 이 절에서는 이들에 대해 간단히 살펴보자.

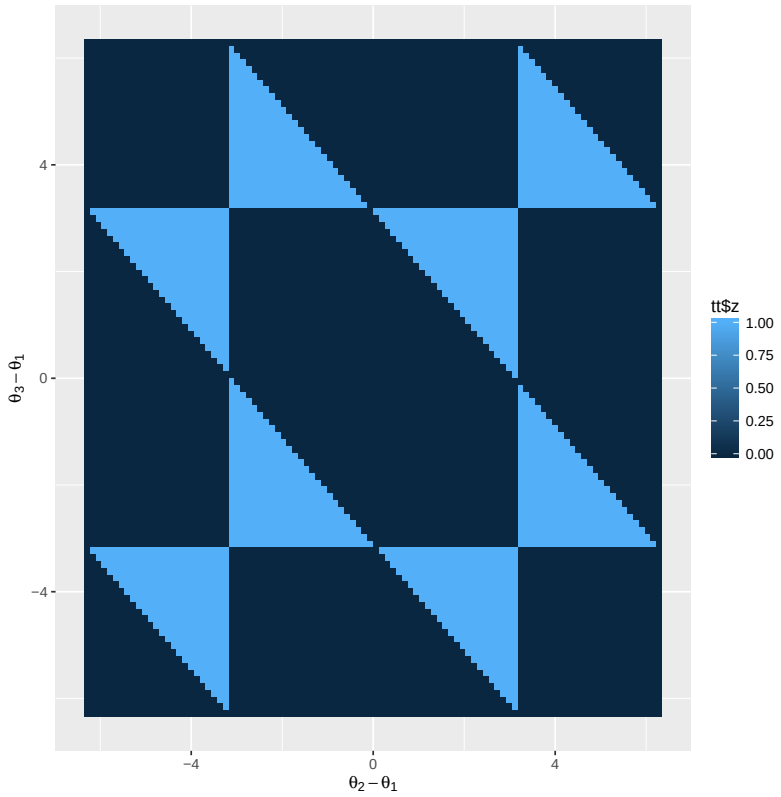


그림 3.1.7. 몬테카를로적분

대수법칙과 중심극한정리를 이해하기 위해서는 우선 확률변수열의 수렴에 대해 알고있어야 한다. 확률변수는 가측공간 $(\Omega, \mathcal{F})$ 상의 가측함수이므로, 확률변수열의 극한이란 함수열의 극한이다. 함수열의 수렴에는 일양수렴 (uniform convergence), 점별수렴 (pointwise convergence), 개수렴 (**槪收斂**, convergence almost surely), 확률수렴 (convergence in probability), 분포수렴 (convergence in distribution)이 있다. 몬테카를로법을 설명하는 경우에는 자주 사용되지 않지만, 재무학에서 중요한 역할을 하는 수렴은 평균수렴 (mean convergence)이다. 이후, 개수렴은 '$a.s.$', 확률수렴은 '$in\ [P]$', 그리고 평균수렴은 '$in\ L^2$' 로 표기하자. 이 수렴들 사이에는 다음 관계가 성립한다.

$$\text{일양수렴} \ \Rightarrow\ \text{점별수렴} \ \Rightarrow \left\{ \begin{array}{c} \text{개수렴} \\ \text{평균수렴} \end{array} \right\} \ \Rightarrow\ \text{확률수렴} \ \Rightarrow\ \text{분포수렴} \tag{3.2.1}$$

중심극한정리는 분포수렴에 관한 정리이고, 약대수법칙 (weak law of large numbers: WLLN)은 확률수렴에 관한 정리이며, 그리고 강대수법칙 (strong law of large numbers: SLLN)은 개수렴에 관한 정리이다. 이들을 설명하기 위해서, 확률변수열 $\{x_n \mid n = 1, 2, \cdots\}$

에 대해 다음과 같이 부분합을 정의하자.

$$S_n \doteq \sum_{j=1}^{n} x(j), \ \ (n = 1, 2, \cdots) \tag{3.2.2}$$

본서에서는 부분합 S_n의 기대값이 유한이라고 가정하자. 다음 정리들이 성립한다.

정리 3.2.1

확률공간 $(\Omega, \mathcal{F}, P)$에서 정의되는 확률변수들 $x_1, x_2, \cdots$이 서로 비상관이고, 또한 다음 식들을 만족하는 상수 M이 존재한다고 가정하자.

$$E(x_n^2) < M, \ \ (n = 1, 2, \cdots)$$

이러한 조건 하에서, 다음 식들이 성립한다.

$$\lim_{n\to\infty} \frac{S_n - E(S_n)}{n} = 0 \ \ in \ \ [P]$$
$$\lim_{n\to\infty} \frac{S_n - E(S_n)}{n} = 0 \quad a.s.$$
$$\lim_{n\to\infty} \frac{S_n - E(S_n)}{n} = 0 \quad in \ \ L^2$$

정리 3.2.2: Khinchine의 약대수법칙

확률공간 $(\Omega, \mathcal{F}, P)$에서 정의되는 확률변수들 $x_1, x_2, \cdots$이 쌍으로 독립이며 또한 동일한 확률분포를 따른다고 하자. 만약 $E(x_1) = \mu(< \infty)$이면, 다음 식이 성립한다.

$$\lim_{n\to\infty} \frac{1}{n} S_n = \mu \ \ in \ \ [P]$$

정리 3.2.3: Chebyshev의 약대수법칙

확률공간 $(\Omega, \mathcal{F}, P)$에서 정의되는 확률변수들 $x_1, x_2, \cdots$이 서로 독립이고, 각 n에 대해서 확률변수 x_n은 확률분포함수 F_n을 따르며, 수열 $\{b_n\}$은 식 $b_n \to \infty$를 만족하고, 또한

다음 식들이 성립한다고 가정하자.

$$\sum_{j=1}^{n} \int_{|x|>b_n} dF(j)(x) = o(1)$$
$$\frac{1}{b_n^2} \sum_{j=1}^{n} \int_{|x|\leq b_n} x^2 dF(j)(x) = o(1)$$

다음과 같이 a_n을 정의하자.

$$a_n \doteq \sum_{j=1}^{n} \int_{|x|\leq b_n} x dF(j)(x)$$

이러한 조건 하에서, $S_n = \sum_{j=1}^{n} x(j)$는 다음 식을 만족한다.

$$\lim_{n\to\infty} \frac{S_n - a_n}{b_n} = 0 \quad in \ [P]$$

정리 3.2.4

확률공간 $(\Omega, \mathcal{F}, P)$에서 정의되고 서로 독립인 확률변수들 $x_1, x_2, \cdots$이 동일한 확률분포를 따르면, 다음 명제들이 성립한다.

$$E(x_1) < \infty \ \Rightarrow \ \lim_{n\to\infty} \frac{1}{n} S_n = E(x_1) \ \ a.s.$$
$$E(x_1) = \infty \ \Rightarrow \ \overline{\lim_{n\to\infty}} \frac{1}{n} |S_n| = \infty \ \ a.s.$$

정리 3.2.5: Kolmogorov의 강대수법칙

확률공간 $(\Omega, \mathcal{F}, P)$에서 정의되고 서로 독립인 확률변수들 $x_1, x_2, \cdots$이 다음 식들을 만족한다고 하자.

$$E(x_n) = \mu_n < \infty, \ \ Var(x_n) = \sigma_n^2 < \infty$$

또한, 다음 식이 성립한다고 하자.

$$\sum_{n=1}^{\infty} \frac{\sigma_n^2}{n^2} < \infty$$

이러한 조건 하에서, 다음 식이 성립한다.

$$\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} [x(i) - \mu(i)] = 0 \;\; a.s.$$

몬테카를로법에서 실험의 시행횟수를 늘리면, 강대수법칙에 의해서 실현값들의 표본평균 $\bar{x}_n \doteq \frac{1}{n}\sum_{j=1}^{n} x(j)$는 원래의 확률분포의 기대값에 수렴한다. 즉, 강대수법칙에서 다음 두 가지 성질들을 알 수 있다. 첫째, 서로 독립인 확률변수열 $\{x_n(\omega)\}$의 표본평균 $\bar{x}_n(\omega)$가 어떤 확률변수 $y(\omega)$에 수렴한다. 둘째, 이 확률변수 $y(\omega)$가 ω에 의존하지 않는 상수 $E(x_1)$이다. 그러나, 일반적으로 확률변수열은 상수가 아닌 어떤 확률변수로 수렴한다. 강대수법칙에 의해서, 다음 결론을 지을 수 있다. 몬테카를로법에서 실험의 횟수를 증가시키면, 실험값들의 표본평균은 모평균에 확률 1로 수렴한다. 반면에, 중심극한정리는 이 표본평균이 정규확률분포를 따르는 확률변수에 분포수렴한다는 것이다. 따라서, 실험의 시행횟수가 충분히 클 때는 표본평균이 정규확률분포를 따른다고 간주할 수 있다. 즉, 중심극한정리를 사용해서, 몬테카를로법으로 구한 표본평균과 모평균의 차이인 오차를 평가할 수 있다. 직관적으로 보면, 실험의 시행횟수 n이 100인 경우와 100,000인 경우 중에서 후자 쪽의 오차가 작다. 중심극한정리는 이 직관이 타당함을 보여준다.

정리 3.2.6: Lindberg-Lévy의 중심극한정리

확률공간 $(\Omega, \mathcal{F}, P)$에서 정의되고 서로 독립인 확률변수들 $\{x_n \mid n = 1, 2, \cdots\}$이 동일한 확률분포를 따르며, $E(x_n) = \mu$이고, $Var(x_n) = \sigma^2$라 가정하자. 이러한 조건 하에서, 다음 식이 성립한다.

$$\frac{\sqrt{n}\,[\bar{x}_n - \mu]}{\sigma} \;\xrightarrow{d}\; \mathcal{N}(0,1)$$

정리 3.2.7: Lindberg-Feller의 중심극한정리

확률공간 $(\Omega, \mathcal{F}, P)$에서 정의되고 서로 독립인 확률변수들 $x_1, x_2, \cdots$이 다음 식들을 만족한다고 하자.

$$E(x_n) = \mu_n < \infty, \;\; Var(x_n) = \sigma_n^2 < \infty, \;\; E(|x_n - \mu_n|^3) < \infty$$

또한, 다음 식이 성립한다고 하자.

$$\lim_{n\to\infty} \frac{\sum_{i=1}^{n} |x(i)-\mu(i)|^3}{[\sum_{j=1}^{n} \sigma(i)^2]^{3/2}} = 0$$

이러한 조건 하에서, 다음 식이 성립한다.

$$\frac{\sum_{j=1}^{n} [x(i)-\mu(i)]}{\sqrt{\sum_{j=1}^{n} \sigma(i)^2}} \xrightarrow{d} \mathcal{N}(0,1)$$

제3.3절 원시몬테카를로법

앞에서도 언급했듯이, 원시몬테카를로법(crude Monte Carlo method)는 다음과 같다.

알고리즘 3.3.1: 원시몬테카를로법

서로 독립인 확률변수열 $\{x_n \mid n = 1, 2, \cdots\}$이 확률변수 x와 동일한 확률분포를 따르며, 확률변수 x의 기대값 $E(x)$가 유한이라고 가정하자. 이 기대값 $E(x)$를 표본평균 $\bar{x}_n \doteq \frac{1}{n}\sum_{j=1}^{n} x(j)$로 근사시킬 수 있다.

다음 절에서 알고리즘 3.3.1의 방법을 원시몬테카를로법이라고 하는 이유를 알게 될 것이다. 알고리즘 3.3.1의 조건 하에서 다음 식들이 성립한다.

$$E(\bar{x}_n) = E\left(\frac{1}{n}\sum_{j=1}^{n} x(j)\right) = \frac{1}{n}\sum_{j=1}^{n} E(x(j)) = E(x) \tag{3.3.1}$$

식 (3.3.1)에서 알 수 있듯이, $\bar{x}_n$은 $E(x)$의 불편추정량이다. 또한, 다음 식들이 성립한다.

$$\begin{aligned} Var(\bar{x}_n) &= Var\left(\frac{1}{n}\sum_{j=1}^{n} x(j)\right) = Cov\left(\frac{1}{n}\sum_{j=1}^{n} x(j), \frac{1}{n}\sum_{k=1}^{n} x_k\right) \\ &= \frac{1}{n^2}\sum_{j=1}^{n}\sum_{k=1}^{n} Cov(x(j), x_k) = \frac{1}{n^2}\sum_{j=1}^{n} Var(x(j)) = \frac{1}{n}Var(x) = \frac{\sigma^2}{n} \end{aligned} \tag{3.3.2}$$

여기서 σ는 확률변수 x의 표준편차이다. 식 (3.3.2)에서 알 수 있듯이, $\bar{x}_n$의 표준편차는 $\sigma/\sqrt{n}$이다. 또한, Lindberg-Lévy중심극한정리에서 알 수 있듯이, 다음 식이 성립한다.

$$\sqrt{n}\,[\bar{x}_n - \mu] \;\xrightarrow{d}\; \mathcal{N}\left(0, \sigma^2\right) \tag{3.3.3}$$

따라서, 신뢰수준이 $1-\alpha$인 신뢰구간은 다음과 같다.

$$\left[\bar{x}_n - z_{1-\frac{\alpha}{2}}\frac{\sigma}{\sqrt{n}}\,,\; \bar{x}_n + z_{1-\frac{\alpha}{2}}\frac{\sigma}{\sqrt{n}}\right] \tag{3.3.4}$$

여기서 z_γ는 표준정규확률분포의 $\gamma-$분위수(quantile)이다. 일반적으로 다음과 같이 신뢰수준이 근사적으로 95%인 2σ규칙($2\sigma-$rule)이 널리 사용된다.

$$\left[\bar{x}_n - 2\frac{\sigma}{\sqrt{n}}\,,\; \bar{x}_n + 2\frac{\sigma}{\sqrt{n}}\right] \tag{3.3.5}$$

사전에 표준편차 σ를 알지 못하므로 다음 추정량 $\bar{\sigma}_n$을 사용한다.

$$\bar{\sigma}_n \doteq \sqrt{\frac{1}{n-1}\sum_{j=1}^{n}\left[x(j) - \bar{x}_n\right]^2} \tag{3.3.6}$$

따라서, 신뢰수준이 $1-\alpha$인 신뢰구간을 다음과 같이 추정할 수 있다.

$$\left[\bar{x}_n - z_{1-\frac{\alpha}{2}}\frac{\bar{\sigma}_n}{\sqrt{n}}\,,\; \bar{x}_n + z_{1-\frac{\alpha}{2}}\frac{\bar{\sigma}_n}{\sqrt{n}}\right] \tag{3.3.7}$$

추정값 $\bar{x}_n$의 정밀도(accuracy)를 말할 때 추정된 상대오차(relative error) $\bar{\sigma}_n/\left[\bar{x}_n\sqrt{n}\right]$를 사용한다. 몬테카를로법에 의한 추정값의 정밀도를 높이기 위해서는 샘플들의 개수 n을 증가시키거나 또는 분산 σ^2를 감소시키는 방법을 적용한다. 컴퓨터를 저렴하게 사용할 수 있는 요즈음에는 시뮬레이션 횟수 n을 늘리는 것이 그다지 어렵지 않다. 그러나, 약간의 수리적 노력을 하면 분산을 아주 감소시킬 수도 있다. 이러한 분산감소법에 대해서는 다음 절에서 다룰 것이다.

예제 3.3.1 다음 정적분을 원시몬테카를로법으로 계산해보자.

$$\theta \doteq \int_0^1 \frac{1}{\sqrt{x}}dx \tag{1}$$

지지대가 $(0,1)$인 일양확률분포에 따르는 확률변수 x에 대해서 다음 식이 성립한다.

$$\theta = E\left(\frac{1}{\sqrt{x}}\right) \tag{2}$$

식 (2)의 우변이 2라는 것을 쉽게 증명할 수 있지만, 지지대가 $(0,1)$인 일양확률분포에서 발생된 일양샘플들 $x_1, x_2, \cdots, x_n$을 사용해서 정적분 θ를 다음과 같이 추정할 수 있다.

$$\hat{\theta} = \frac{1}{n}\sum_{j=1}^{n}\frac{1}{\sqrt{x(j)}} \tag{3}$$

다음 식이 성립한다.

$$Var\left(\frac{1}{\sqrt{x}}\right) = E\left(\frac{1}{x}\right) - \left[E\left(\frac{1}{\sqrt{x}}\right)\right]^2 = \int_0^1 \frac{1}{x}dx - 2^2 \tag{4}$$

그러나, $E(1/x)$은 ∞이므로, 식 (4)의 우변은 무한대이다. 즉, 추정량 $\hat{\theta}$는 좋은 것이 아니다.

변수변환 $y = \sqrt{x}$를 적용하면, Jacobian이 $1/[2\sqrt{x}]$이므로 다음 식이 성립한다.

$$\theta = 2\int_0^1 dy \tag{5}$$

식 (5)의 우변은 지지대가 $(0,1)$인 일양확률분포에 따르는 확률변수 y에 대해서 상수함수 1의 기대값을 구하는 것이다. 따라서, θ는 2이고 분산이 0이다. 즉, 어떤 일양샘플들이 추출되느냐에 상관없이 추정값이 2이다.

지금까지 내용을 확인하기 위해서, 다음 MATLAB프로그램 CrudeMC101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
% Filename: CrudeMC101.m
% Crude Monte Carlo Simulation
% Programmed by CBS
% ------------------------------------------------------------------------
clear all, clf, close all
rand('twister',5489);   % Use twister or state for seed.
LengSim = 2^14;
NoSim = 1000;
urn = rand(LengSim,NoSim);
theta1 = mean(1./sqrt(urn));
Mtheta1 = mean(theta1)
Vtheta1 = var(theta1)
theta2 = 2*mean(ones(LengSim,NoSim));
Mtheta2 = mean(theta2)
Vtheta2 = var(theta2)
% Plotting
```

```
histfit(theta1)
set(gca,'fontsize',11,'fontweigh','bold')
saveas(gcf,'CrudeMC101','epsc')
save('CrudeMC101','theta1','theta2')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램 CrudeMC101.m을 실행하면, 식 (2)와 식 (5)를 바탕으로 하는 원시몬테카를로법이 실행된다. 각 시뮬레이션에서 $2^{14} = 16384$개 샘플들을 생성하고, 이러한 시뮬레이션을 1000번 한다. 첫 번째 원시몬테카를로법에서는 이 정적분의 추정값들의 표본평균이 1.9992이고 분산은 $7.41 \cdot 10^{-4}$이다. 반면에, 두 번째 원시몬테카를로법에서는 이 정적분의 추정값들의 표본평균이 0이고 분산도 0이다. 이 MATLAB프로그램을 실행하면, 그림 3.3.1를 출력한다. 그림 3.3.1에서 알 수 있듯이, 첫 번째 원시몬테카를로법에 의한 분산은 상당히 크다. 즉, 이 추정값의 표본분포는 꼬리가 무겁다.

이 예제에서 알 수 있듯이, 피적분함수를 상수에 가깝게 변환하면 몬테카를로법에 의한 적분추정값의 정밀도를 높일 수 있다. ■

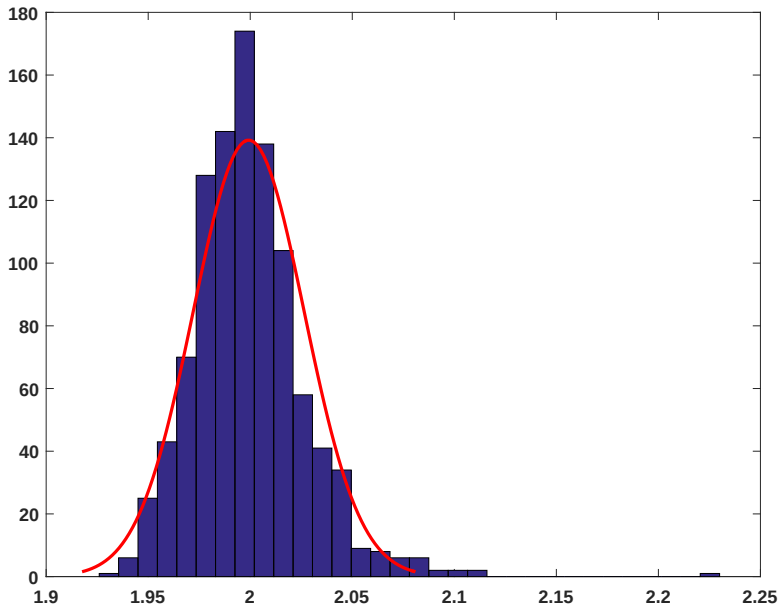


그림 3.3.1. 원시몬테카를로적분 1 (MATLAB)

예제 3.3.2 R언어를 사용해서 예제 3.3.1의 문제를 풀기 위해, 다음 R프로그램 CrudeMC101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
```

```
# Filename: CrudeMC101R.R
# Crude Monte Carlo Simulation
# Programmed by CBS
# ------------------------------------------------------------------------
# Generating Random Numbers
set.seed(11)
LengSim <- 2^14
nSim <- 1000
urn <- runif(LengSim*nSim)
( dim(urn) = c(LengSim,nSim) )
theta1 <- colMeans(1/sqrt(urn))
( Mtheta1 <- mean(theta1) )
( Vtheta1 <- var(theta1) )
theta2 <- 2*colMeans(matrix(1,LengSim,nSim))
( Mtheta2 <- mean(theta2) )
( Vtheta2 <- var(theta2) )

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages('latex2exp')
library(latex2exp)
setEPS()
plot.new()
postscript('CrudeMC101R.eps')  # Start to save figure
RVdata <- data.frame(nSim,theta1,theta2)
ggplot(RVdata,aes(x=theta1)) +
 geom_histogram(binwidth=0.01,col="black",fill="yellow") +
 stat_function(
     fun = function(x, mean, sd, n, bw){
           dnorm(x=x, mean=mean, sd=sd)*n*bw
     },
     args = c(mean=Mtheta1, sd=sqrt(Vtheta1), n=nSim, bw=0.01),
     lwd=1.2, col="dark red"
 ) +
 xlab(TeX('$\\theta_{1}$')) +
 xlim(c(1.9,2.25))
dev.off()            # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램 CrudeMC101R.m을 실행하면, 각 시뮬레이션에서 $2^{14} = 16384$개 샘플들을 생성하고, 이러한 시뮬레이션을 1000번 한다. 첫 번째 원시몬테카를로법에서는 이 정적분의 추정값들의 표본평균이 1.9989이고 분산은 $6.94 \cdot 10^{-4}$이다. 반면에, 두 번째 원시몬테카를로법에서는 이 정적분의 추정값들의 표본평균이 0이고 분산도 0이다. 이 R프로그램을 실행하면, 그림 3.3.2를 출력한다. 그림 3.3.2에서 알 수 있듯이, 첫 번째 원시몬테카를로법에 의한 분산은 상당히 크다. 즉, 이 추정값의 표본분포는 꼬리가 무겁다.

이 예제에서 알 수 있듯이, 피적분함수를 상수에 가깝게 변환하면 몬테카를로법에 의한 적분추정값의 정밀도를 높일 수 있다. ■

예제 3.3.3 예제 3.1.3에서는 Buffon바늘문제의 부산물로 원주율 π값을 수치적으로 구했다.

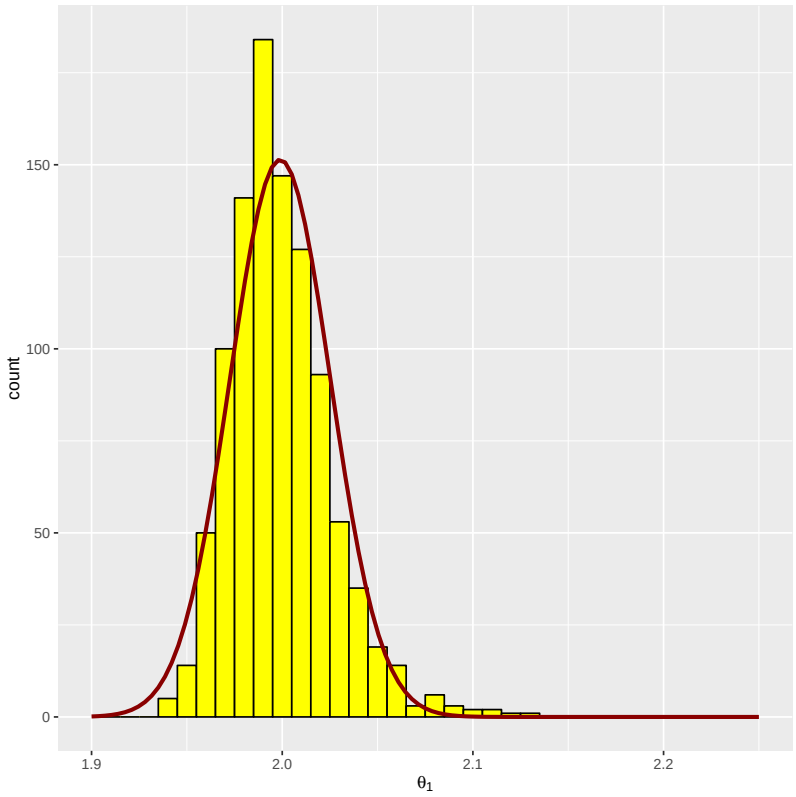


그림 3.3.2. 원시몬테카를로적분 1 (R)

이 예제에서는 다른 원시몬테카를로법들을 적용해서 π 값을 구해보자.

첫째, 히트앤미스법 (hit and miss method) 를 사용해서 π 값을 구해보자. 원점을 중심으로하고 반경 1 인 원의 제1사분면 부분의 면적은 $\pi/4$ 이다. 즉, 다음 식이 성립한다.

$$\frac{\pi}{4} = \int_0^1 \int_0^1 1\left(u^2 + v^2 \leq 1\right) dudv \tag{1}$$

만약 $(u_1, v_1), (u_2, v_2), \cdots, (u_n, v_n)$ 을 서로 독립이고 지지대가 $(0,1)$ 인 일양샘플들의 쌍들이라고 하면, 이들은 정사각형 $(0,1) \times (0,1)$ 상에서 일양분포한다. 이 n 쌍 중에서 식 $u(i)^2 + v(i)^2 < 1$ 을 만족하는 개수를 w 라고 하면, π 를 다음과 같이 추정할 수 있다.

$$\tilde{\pi} = 4\frac{w}{n} \tag{2}$$

확률변수 w 는 모수들이 n 과 $\pi/4$ 인 이항확률분포를 따르므로, 추정량 $\tilde{\pi}$ 의 분산은 다음과 같다.

$$Var\left(\tilde{\pi}\right) = 16\frac{Var(w)}{n^2} = \frac{16}{n}\frac{\pi}{4}\left[1 - \frac{\pi}{4}\right] = \frac{\pi[4-\pi]}{n} = \frac{2.6968}{n} \tag{3}$$

둘째, 다음 식을 이용하기로 하자.

$$\frac{\pi}{4} = \int_0^1 \sqrt{1-u^2}du \tag{4}$$

지지대가 $(0,1)$인 일양샘플들을 $u_1, u_2, \cdots, u_n$이라고 하면, π를 다음과 같이 추정할 수 있다.

$$\hat{\pi} = 4 \cdot \frac{1}{n}\sum_{j=1}^{n} \sqrt{1-u(j)^2} \tag{5}$$

다음 식들이 성립한다.

$$Var\left(\sqrt{1-u^2}\right) = E\left(1-u^2\right) - \left[\frac{\pi}{4}\right]^2 = 1 - \frac{1}{3} - \frac{\pi^2}{16} = 0.0498 \tag{6}$$

따라서, 추정량 $\hat{\pi}$의 분산은 다음과 같다.

$$Var(\hat{\pi}) = \frac{16}{n}Var(\sqrt{1-u^2}) = \frac{16 \cdot 0.0498}{n} = \frac{0.7971}{n} \tag{7}$$

식 (3)과 식 (7)에서 알 수 있듯이 두 번째 방법이 첫 번째 방법보다 $[2.6968/n]/[0.7971/n] = 3.3833$ 배만큼 효율이 좋다. 지금까지 내용을 확인하기 위해서, 다음 MATLAB프로그램 CrudeMC102.m을 실행해 보자.

```
% ------------------------------------------------------------------------
% Filename: CrudeMC102.m
% Crude Monte Carlo Simulation for pi
% Programmed by CBS
% ------------------------------------------------------------------------
clear all, clf, close all
rand('twister',5489);   % Use twister or state for seed.
for kk = 1:30
    u = rand(1000,2);
    dum11 = ( u(:,1).^2 + u(:,2).^2 <= 1 );
    dum12(kk) = 4*sum(dum11)/1000;
    dum21 = sqrt(1-u(:,1).^2 );
    dum22(kk) = 4*sum(dum21)/1000;
end
pi1 = cumsum(dum12)./(1:30)
pi2 = cumsum(dum22)./(1:30)
% Plotting
Nobs = (1:1:30)*1000;
plot(Nobs,pi1,'r-',Nobs,pi2,'k--',[1 30000],[pi pi],'LineWidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
legend('Mehtod 1','Method 2','True value','location','NE')
xlabel('\bf Number of Samples','Fontsize',12');
ylabel('\bf Estimates','Fontsize',12');
saveas(gcf,'CrudeMC102','epsc')
save('CrudeMC102','pi1','pi2')
```

```
% End of program
% --------------------------------------------------------------------------
```

이 MATLAB프로그램 CrudeMC102.m을 실행하면, 식 (1)과 식 (5)를 바탕으로 하는 원시몬테카를로법이 실행된다. 샘플들의 개수가 1000개인 시뮬레이션부터 시작해서, 샘플들이 1000개 씩 증가할 때마다 시뮬레이션을 한 번하고, 샘플들이 30000개인 경우 마지막 시뮬레이션을 한다. 이 MATLAB프로그램을 실행하면, 그림 3.3.3를 출력한다. 그림 3.3.3에서 알 수 있듯이, 두 번째 원시몬테카를로법에 의한 추정값이 첫 번째 원시몬테카를로법에 의한 추정값보다 빨리 참값에 수렴한다. ■

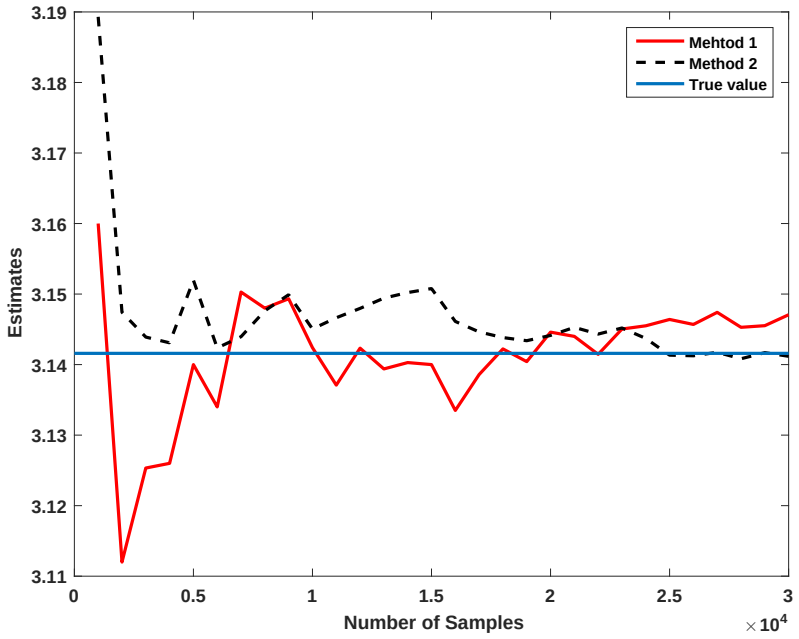


그림 3.3.3. 원시몬테카를로적분 2 (MATLAB)

예제 3.3.4 R언어를 사용해서 예제 3.3.3의 문제를 풀기 위해, 다음 R프로그램 CrudeMC102R.R을 실행해 보자.

```
# --------------------------------------------------------------------------
# Filename: CrudeMC102R.R
# Crude Monte Carlo Simulation
# Programmed by CBS
# --------------------------------------------------------------------------
# Generating Random Numbers and Simulations
set.seed(11)
dum12 <- rep(0,30)
dum22 <- rep(0,30)
for (kk in 1:30){
```

```
    u <-runif(1000*2)
    dim(u) <- c(1000,2)
    dum11 = 1*( u[,1]^2 + u[,2]^2 <= 1 )
    dum12[kk] = 4*sum(dum11)/1000
    dum21 = sqrt(1-u[,1]^2 )
    dum22[kk] = 4*sum(dum21)/1000
}
pi1 <- cumsum(dum12)/c(1:30)
pi2 <- cumsum(dum22)/c(1:30)
# Plotting
# install.packages("ggplot2")
library(ggplot2)
setEPS()
plot.new()
postscript('CrudeMC102R.eps')  # Start to save figure
nSim <- c(1:1:30)*1000
RVdata <- data.frame(nSim,pi1,pi2)
ggplot(RVdata,aes(x=nSim,y=pi1)) +
 geom_line(col="red",lwd=1.2,linetype=1)+
 geom_line(aes(x=nSim,y=pi2),col="black",lwd=1.2,linetype=2) +
 geom_hline(yintercept=pi,col="blue",lwd=1) +
 xlab("Number of Samples") + ylab("Estimate")
dev.off()            # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램 CrudeMC102R.m을 실행하면, 원시몬테카를로법이 실행된다. 샘플들의 개수가 1000개인 시뮬레이션부터 시작해서, 샘플들이 1000개 씩 증가할 때마다 시뮬레이션을 한 번하고, 샘플들이 30000개인 경우 마지막 시뮬레이션을 한다. 이 R프로그램을 실행하면, 그림 3.3.4를 출력한다. 그림 3.3.4에서 알 수 있듯이, 두 번째 원시몬테카를로법에 의한 추정값이 첫 번째 원시몬테카를로법에 의한 추정값보다 빨리 참값에 수렴한다. ■

예제 3.3.5 다음 정적분을 원시몬테카를로법으로 계산해보자.

$$\theta = \int_1^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}z^2\right) dz \tag{1}$$

첫째, 표준정규샘플들 $z_1, z_2, \cdots, z_n$들 중에서 1이상인 것들의 개수를 w라고 하면, θ를 다음과 같이 추정할 수 있다.

$$\tilde{\theta} = \frac{w}{n} \tag{2}$$

다음 식이 성립함을 쉽게 알 수 있다.

$$Var\left(\tilde{\theta}\right) = \frac{\theta[1-\theta]}{n} \tag{3}$$

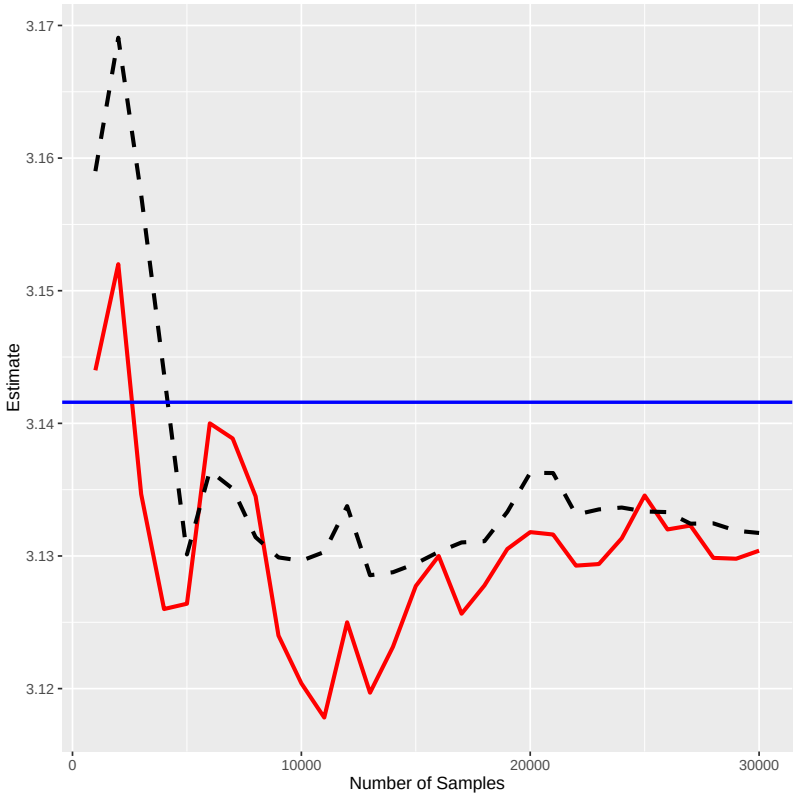


그림 3.3.4. 원시몬테카를로적분 2 (R)

둘째, 다음 식을 이용하기로 하자.

$$\theta = \frac{1}{2} Pr(|z| > 1) \tag{4}$$

여기서 z는 표준정규확률변수이다. 따라서, 표준정규샘플들 $z_1, z_2, \cdots, z_n$들 중에서 절대값이 1보다 큰 것들의 개수를 v라고 하면, θ를 다음과 같이 추정할 수 있다.

$$\check{\theta} = \frac{v}{2n} \tag{5}$$

확률변수 v는 모수들이 n과 2θ인 이항확률분포를 따르므로, 추정량 $\check{\theta}$의 분산은 다음과 같다.

$$Var\left(\check{\theta}\right) = \frac{2\theta[1-2\theta]}{4n} = \frac{\theta[1-2\theta]}{2n} \tag{6}$$

셋째, 다음 식을 이용하기로 하자.

$$\theta = \frac{1}{2} - \int_0^1 \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}z^2\right) dz \tag{7}$$

지지대가 $(0,1)$인 일양샘플들을 $u_1, u_2, \cdots, u_n$이라고 하면, θ를 다음과 같이 추정할 수 있다.

$$\hat{\theta} = \frac{1}{2} - \frac{1}{\sqrt{2\pi}} \cdot \frac{1}{n} \sum_{i=1}^{n} \exp\left(-\frac{1}{2}u(i)^2\right) \tag{8}$$

다음 식들이 성립한다.

$$\begin{aligned} & Var\left(\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}u^2\right)\right) \\ &= E\left(\frac{1}{2\pi} \exp(-u^2)\right) - \left[\frac{1}{2} - \theta\right]^2 \\ &= \int_0^1 \frac{1}{2\pi} \exp(-u^2) du - \left[\frac{1}{2} - \theta\right]^2 \end{aligned} \tag{9}$$

따라서, 추정량 $\hat{\theta}$의 분산은 다음과 같다.

$$Var\left(\check{\theta}\right) = \frac{1}{n}\left\{\int_0^1 \frac{1}{2\pi} \exp(-u^2) du - \left[\frac{1}{2} - \theta\right]^2\right\} \tag{10}$$

컴퓨터를 사용해서, 다음 식이 성립함을 알 수 있다.

$$\theta = 0.1587 \tag{11}$$

따라서, 각 추정량의 분산은 다음과 같다.

$$Var\left(\tilde{\theta}\right) = \frac{0.1335}{n} \tag{12}$$

$$Var\left(\check{\theta}\right) = \frac{0.0542}{n} \tag{13}$$

$$Var\left(\hat{\theta}\right) = \frac{0.002345}{n} \tag{14}$$

식 (12)와 식 (13)에서 알 수 있듯이, 추정량 $\tilde{\theta}$에 비해서 추정량 $\check{\theta}$가 2.4648배 효율적이다. 그러나, 식 (12)와 식 (14)에서 알 수 있듯이, 추정량 $\tilde{\theta}$에 비해서 추정량 $\hat{\theta}$가 56.9227배 효율적이다. 추정량 $\hat{\theta}$을 계산하는 데는 지수함수 등을 계산하는데 시간이 걸리긴 하지만, 추정량의 효율이 아주 좋다.

지금까지 내용을 확인하기 위해서, 다음 MATLAB프로그램 Crude_MC103.m을 실행해 보자.

```
% --------------------------------------------------------------------
```

```
% Filename: CrudeMC103.m
% Crude Monte Carlo Simulation for Pr(N(0,1) >= 1)
% Programmed by CBS
% ------------------------------------------------------------------------
clear all, clf, close all
rand('twister',5489);   % Use twister or state for seed.
Kmax = 30
for kk = 1:Kmax
    z = normrnd(0,1,1000,1);
    dum11 = ( z(:) >= 1 );
    dum12(kk) = sum(dum11)/1000;
    dum21 = ( abs(z(:)) >= 1 );
    dum22(kk) = sum(dum21)/(2*1000);
    u = rand(1000,1);
    dum31 = normpdf(u);
    dum32(kk) = sum(dum31)/1000;
end
MC1 = cumsum(dum12)./(1:Kmax);
MC2 = cumsum(dum22)./(1:Kmax);
MC3 = 0.5 - cumsum(dum32)./(1:Kmax);
% Plotting
Value = 1 - normcdf(1,0,1)
Nobs = (1:1:Kmax)*1000;
plot([1 30000],[Value Value],'k:',Nobs,MC1,'r-',Nobs,MC2,'g--', ...
      Nobs,MC3,'b:','LineWidth',1.8)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[0.15,0.17])
legend('True value','Mehtod 1','Method 2', ...
        'Method 3','location','SE')
xlabel('\bf Number of Samples','Fontsize',12');
ylabel('\bf Estimates','Fontsize',12');
saveas(gcf,'CrudeMC103','epsc')
save('Crude_MC103','MC1','MC2','MC3')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램 Crude_MC103.m을 실행하면, 식 (1), 식 (4)와 식 (7)를 바탕으로 하는 원시몬테카를로법이 실행된다. 샘플들의 개수가 1000개인 시뮬레이션부터 시작해서, 샘플들이 1000개 씩 증가할 때마다 시뮬레이션을 한 번하고, 샘플들이 30000개인 경우 마지막 시뮬레이션을 한다. 이 MATLAB프로그램을 실행하면, 그림 3.3.5를 출력한다. 그림 3.3.5에서 알 수 있듯이, 세 번째 몬테카를로법에 의한 추정값이 첫 번째 원시몬테카를로법 또는 두 번째 원시몬테카를로법에 의한 추정값보다 빨리 참값에 수렴한다. ■

예제 3.3.6 R언어를 사용해서 예제 **??**의 문제를 풀기 위해, 다음 R프로그램 CrudeMC103R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: CrudeMC103R.R
# Crude Monte Carlo Simulation  for P(N(0,1) >= 1)
# Programmed by CBS
# ------------------------------------------------------------------------
```

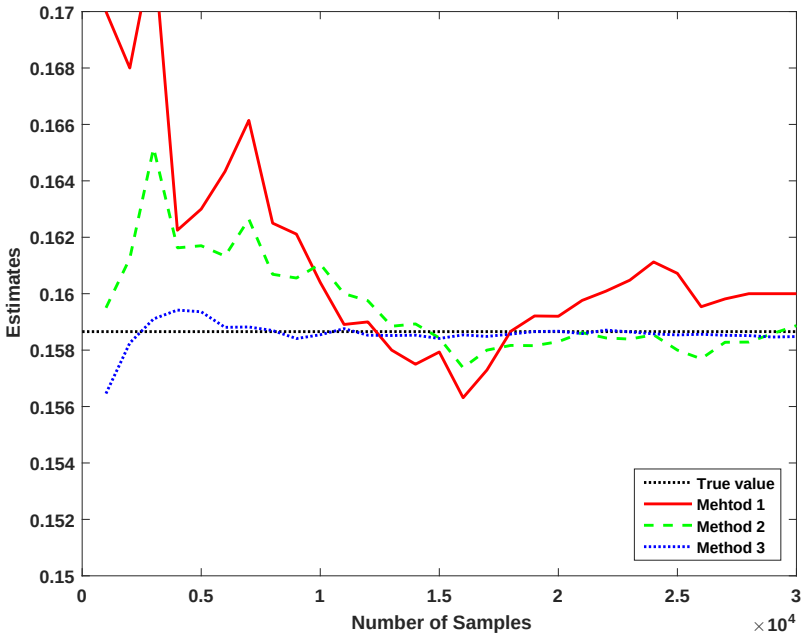


그림 3.3.5. 원시몬테카를로적분 3 (MATLAB)

```r
# Generating Random Numbers and Simulations
set.seed(11)
Valuee <- 1 -pnorm(1)
Kmax <- 30
dum12 <- rep(0,Kmax)
dum22 <- rep(0,Kmax)
dum32 <- rep(0,Kmax)
for (kk in 1:Kmax){
    z <- rnorm(1000)
    dum11 <- ( z >= 1 )
    dum12[kk] <- sum(dum11)/1000
    dum21 <- ( abs(z) >= 1 )
    dum22[kk] <- sum(dum21)/(2*1000)
    u <- runif(1000)
    dum31 <- dnorm(u)
    dum32[kk] <- sum(dum31)/1000
}
MC1 <- cumsum(dum12)/(1:Kmax)
MC2 <- cumsum(dum22)/(1:Kmax)
MC3 <- 0.5 - (cumsum(dum32)/(1:Kmax))

# Plotting
# install.packages("ggplot2")
library(ggplot2)
setEPS()
plot.new()
postscript('CrudeMC103R.eps')  # Start to save figure
nSim <- c(1:1:30)*1000
RVdata <- data.frame(nSim,MC1,MC2,MC3)
ggplot(RVdata,aes(x=nSim,y=MC1)) +
 geom_line(col="red",lwd=1.2,linetype=1)+
 geom_line(aes(x=nSim,y=MC2),col="green",lwd=1.2,linetype=2) +
 geom_line(aes(x=nSim,y=MC3),col="blue",lwd=1.2,linetype=3) +
 geom_hline(yintercept=Valuee,col="black",lwd=1) +
 xlab("Number of Samples") + ylab("Estimate") +
 ylim(0.15,0.17)
```

```
dev.off()             # End to save figure
# ------------------------------------------------------------------------
```

이 R프로그램 CrudeMC103R.m을 실행하면, 샘플들의 개수가 1000개인 시뮬레이션부터 시작해서, 샘플들이 1000개 씩 증가할 때마다 시뮬레이션을 한 번하고, 샘플들이 30000개인 경우 마지막 시뮬레이션을 한다. 이 R프로그램을 실행하면, 그림 3.3.6를 출력한다. 그림 3.3.6에서 알 수 있듯이, 세 번째 몬테카를로법에 의한 추정값이 첫 번째 원시몬테카를로법 또는 두 번째 원시몬테카를로법에 의한 추정값보다 빨리 참값에 수렴한다. ■

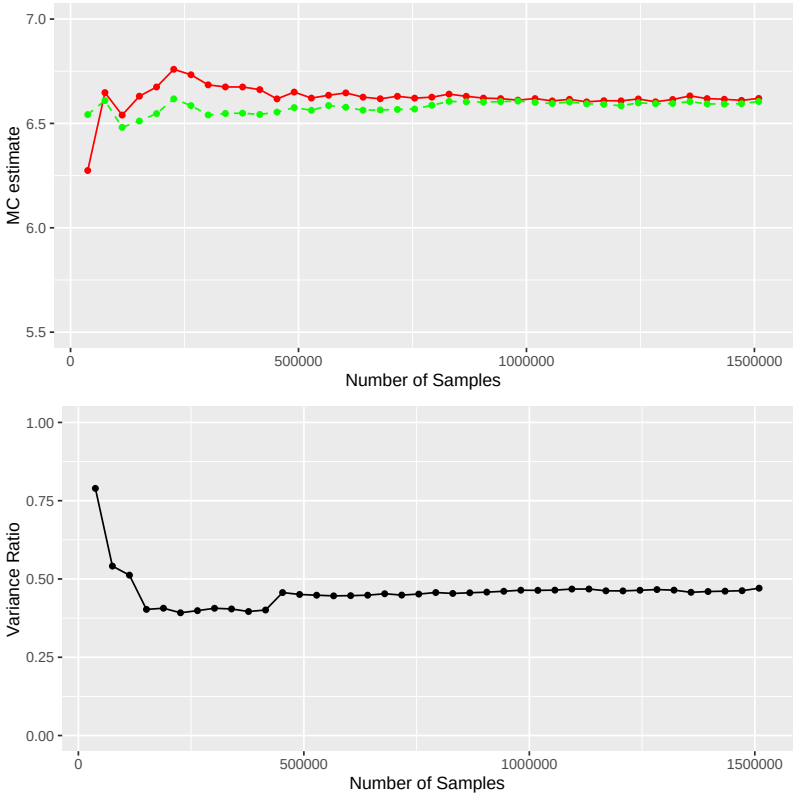


그림 3.3.6. 원시몬테카를로적분 3 (R)

예제 3.3.7 다리네트워크문제(bridge network problem)을 살펴보자. 이 문제는 Kroese & Taimre & Botev [26, p. 348]에서 인용한 것이다. 우선 그림 3.3.7을 그리기 위해서 다음 MATLAB프로그램 BridgeNetwork101.m을 실행하라.

```
% ------------------------------------------------------------------------
%  Filename: BridgeNetwork101.m
% Draw Bridge Network by Kroese (2511)
%  Programmed by CBS
%-------------------------------------------------------------------------
```

```
clear all, close all
% Plotting
plot( [ 0 1 2 1 0],[ 0 1 0 -1 0 ],'k-d','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','color','w')
hold on
plot( [ 1 1 ],[ 1 -1 ],'k-d','linewidth',2)
hold off
axis off
axis( [ - 0.3 2.3 -1.3 1.3 ])
text(-0.23,0,'\bf A','fontsize',25)
text(2.0,0,'\bf B','fontsize',25)
text(0.18,0.5,'\bfx_{1}','fontsize',25)
text(0.2,-0.5,'\bfx_{2}','fontsize',25)
text(1.03,0.0,'\bfx_{3}','fontsize',25)
text(1.6,0.5,'\bfx_{4}','fontsize',25)
text(1.6,-0.5,'\bfx_{5}','fontsize',25)
saveas(gcf,'BridgeNetwork101','epsc')
% End of program
% --------------------------------------------------------------------------
```

■

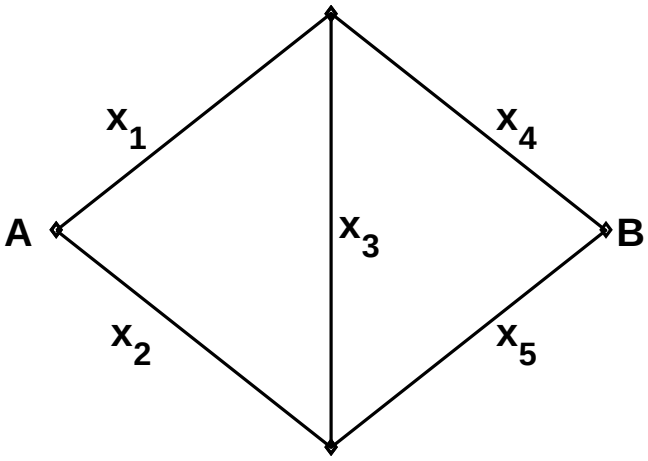


그림 3.3.7. 다리네트워크

예제 3.3.8 이 다리네트워크문제는 그림 3.3.7의 점 A에서 점 B까지 최단거리의 기대값을 구하는 것이다. 제 i번째 다리의 길이 $x(i)$는 지지대가 $(0, a(i))$인 일양확률분포를 따른다. 이 길이들은 서로 독립이며, $a_1 = 1$, $a_2 = 2$, $a_3 = 3$, $a_4 = 1$, 그리고 $a_5 = 2$이다. 각 $i(= 1, 2, 3, 4, 5)$에 대해서 $x(i) = a(i)u(i)$라고 하면, $\{u(i)\}$는 지지대가 $(0, 1)$이고 서로 독립인 일양확률변수들이다. 다음 함수를 정의하자.

$$h(\boldsymbol{u}) \doteq \min\{a_1u_1 + a_4u_4, a_1u_1 + a_3u_3 + a_5u_5, a_2u_2 + a_3u_3 + a_4u_4, a_2u_2 + a_5u_5\} \qquad (1)$$

여기서 $\boldsymbol{u} = [u_1, u_2, u_3, u_4, u_5]^t$ 이다. 우리의 목적은 최소평균거리 $l \doteq E\left(h(\boldsymbol{u})\right)$ 을 구하는 것이다. 이론적으로 계산한 값은 다음과 같다.

$$l = \frac{1339}{1440} = 0.9298611111\cdots \tag{2}$$

원시몬테카를로법에 의한 l 의 추정식은 다음과 같다.

$$\hat{l} = \frac{1}{n}\sum_{j=1}^{n} h\left(\boldsymbol{u}(j)\right) \tag{3}$$

여기서 $\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_n$ 은 지지대가 $[0,1]^5$ 이고 상관계수행렬이 I_5 인 일양샘플벡터들이다.

식 (3)을 사용해서 l 을 추정하기 위해서, 다음 MATLAB프로그램 BridgeNetworkCrudeMC101.m 을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: BridgeNetworkCrudeMC101.m
%  Crude Monte Carlo Simulation of Bridge Network Problem
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all
rand('twister',5489)
hu = []
numadd = 1000
for ii = 1:20
    ndum(ii) = numadd*ii;
    SRndum(ii) = sqrt(ndum(ii));
    xx = rand(numadd,5);
    x1 = 1*xx(:,1); x2 = 2*xx(:,2); x3 = 3*xx(:,3);
    x4 = 1*xx(:,4); x5 = 2*xx(:,5);
    Path1 = x1 + x4;
    Path2 = x1 + x3 + x5;
    Path3 = x2 + x3 + x4;
    Path4 = x2 + x5;
    hudum = min([Path1,Path2,Path3,Path4],[],2);
    hu = [ hu; hudum ];
    meanl(ii) = mean(hu)
    stdl(ii) = std(hu)
    RelativeError(ii) = stdl(ii)/sqrt(ndum(ii))/meanl(ii)
end
% Plotting
subplot(2,1,1)
plot(ndum,meanl,'r-*',ndum,meanl-2*stdl./SRndum,'b--', ...
     ndum,meanl+2*stdl./SRndum,'b--','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[0.8,1.2])
legend('Estimate','Confidence Interval','location','NE')
xlabel('\bf Number of Random Samples')
ylabel('\bf Estimate and CI')
hold off
subplot(2,1,2)
plot(ndum,RelativeError,'k-*','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[0 0.015])
xlabel('\bf Number of Random Samples')
```

```
ylabel('\bf Relative Error')
hold off
saveas(gcf,'BridgeNetworkCrudeMC101','epsc')
save('BridgeNetworkCrudeMC101','meanl','RelativeError')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 원시몬테카를로법을 사용해서, 다리네트워크의 최소평균거리 $l = E\left(h(\boldsymbol{u})\right)$를 추정한다. 샘플벡터들의 개수를 1000개부터 1000개씩 20000개까지 늘려가면서, 원시몬테카를로법의 추정값 $\hat{l}$을 계산한다. 이렇게 계산된 추정값들과 신뢰구간들 그리고 상대오차들이 그림 3.3.7에 그려져 있다. 예상대로, 샘플벡터들의 개수가 증가됨에 따라 표본평균은 참값 1339/1440에 수렴하고 신뢰구간은 점차 짧아진다. 또한, 상대오차도 감소한다. 관찰점들의 개수가 20000인 경우 추정값, 표준편차, 그리고 추정된 상대오차는 각각 다음과 같다.

$$\hat{l}_{20000}^{Crude} = 0.9295, \quad \hat{\sigma}_{20000}^{Crude} = 0.3972, \quad RE_{20000}^{Crude} = 0.0030 \tag{4}$$

■

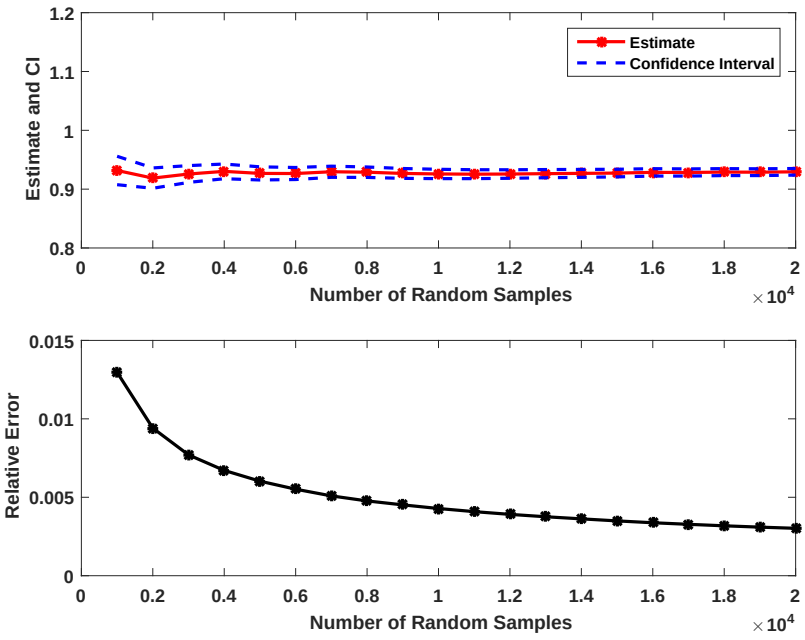


그림 3.3.8. 원시몬테카를로법

예제 3.3.9 R언어를 사용해서 예제 3.3.8의 문제를 풀기 위해, 다음 R프로그램 BridgeNet-

workCrudeMC101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------------
# Filename: BridgeNetworkCrudeMC101R.R
# Crude Monte Carlo Simulation of Bridge Network Problem
# Programmed by CBS
# ------------------------------------------------------------------------------
# Generating Random Numbers and Simulations
set.seed(11)
hu <- matrix(, nrow=0, ncol=1)
NoSim <- 20
numadd <- 1000
ndum <- rep(0,NoSim)
SRndum <- ndum
meanl <- ndum
sdl <- ndum
RelativeError <- ndum
for (ii in 1:NoSim){
    ndum[ii] <- numadd*ii
    SRndum[ii] <- sqrt(ndum[ii])
    xx <- runif(numadd*5)
    dim(xx) <- c(numadd,5)
    x1 = 1*xx[,1]; x2 = 2*xx[,2]; x3 = 3*xx[,3]
    x4 = 1*xx[,4]; x5 = 2*xx[,5]
    Path1 <- x1 + x4
    Path2 <- x1 + x3 + x5
    Path3 <- x2 + x3 + x4
    Path4 <- x2 + x5
    dumPath <- cbind(Path1,Path2,Path3,Path4)
    hudum <- apply(dumPath,1,min)
    hu <- 1*rbind(hu,hudum)
    meanl[ii] <- mean(hu)
    sdl[ii] <- sd(hu)
    RelativeError[ii] = sdl[ii]/sqrt(ndum[ii])/meanl[ii]
}

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('BridgeNetworkCrudeMC101R.eps')  # Start to save figure
nSim <- 1000*c(1:1:20)
RVdata <- data.frame(nSim,meanl,sdl,RelativeError)
plotMC <- ggplot(RVdata,aes(x=nSim,y=meanl)) +
          geom_point(shape=16,col="black",lwd=1.7)+
          geom_line(col="red") +
          geom_hline(yintercept=0.9299,col="dark blue",lwd=1) +
          geom_line(aes(x=nSim,y=meanl+2*sdl/SRndum),
                    col="blue",lwd=1.2,linetype=2) +
          geom_line(aes(x=nSim,y=meanl-2*sdl/SRndum),
                    col="blue",lwd=1.2,linetype=2) +
          ylim(0.8,1.2) +
          xlab("Number of Samples") + ylab("Estimate")
plotEr <- ggplot(RVdata,aes(x=nSim,y=RelativeError)) +
          geom_point(shape=16,col="black",lwd=2.0) +
          geom_line(col="black") +
          geom_hline(yintercept=0.0,col="dark blue",lwd=1) +
          ylim(0,0.015) +
```

```
          xlab("Number of Samples") + ylab("Relative Error")
pushViewport(viewport(layout=grid.layout(2,1)))
print(plotMC, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plotEr, vp=viewport(layout.pos.row=2, layout.pos.col=1))
dev.off()            # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 원시몬테카를로법을 사용해서, 다리네트워크의 최소평균거리 $l = E\left(h(\boldsymbol{u})\right)$를 추정한다. 샘플벡터들의 개수를 1000개부터 1000개씩 20000개까지 늘려가면서, 원시몬테카를로법의 추정값 $\hat{l}$을 계산한다. 이렇게 계산된 추정값들과 신뢰구간들 그리고 상대오차들이 그림 **??**에 그려져 있다. 예상대로, 샘플벡터들의 개수가 증가됨에 따라 표본평균은 참값 1339/1440에 수렴하고 신뢰구간은 점차 짧아진다. 또한, 상대오차도 감소한다. 관찰점들의 개수가 20000인 경우 추정값, 표준편차, 그리고 추정된 상대오차는 각각 다음과 같다.

$$\hat{l}_{20000}^{Crude} = 0.9295, \quad \hat{\sigma}_{20000}^{Crude} = 0.3972, \quad RE_{20000}^{Crude} = 0.0030 \tag{4}$$

■

제3.4절 분산감소법

원시몬테카를로법은 샘플를 이용해서 정적분을 추정하는 것이다. 그러나, 이 추정량의 분산이 크다면, 추정된 정적분의 신뢰성이 떨어진다. 따라서, 적절한 방법을 사용해서 추정량의 분산을 감소시킨다면, 비교적 작은 개수의 샘플들로부터 정밀도가 좋은 추정값을 구할 수도 있다. 이러한 방법을 총칭해서 분산감소법(variance reduction method)이라 부른다. 이 절에서는 분산감소법에 대해서 간단히 설명하고자 한다. 이에 대한 좀 더 자세한 내용, 특히 재무학이나 경제학에 적용하는 예제에 관해서는 최병선 [2, 제4.5절]을 참조하라.

이 절에서는 확률밀도함수가 $f(x)$인 확률변수 x의 기대값 $E(x)$과 분산 $Var(x)$를 각각 θ와 σ^2로 표기하자. 확률밀도함수 $f(x)$에서 생성한 샘플들 $x_1, x_2, \cdots, x_n$의 표본평균 $\bar{x}_n = \frac{1}{n}\sum_{j=1}^{n} x(j)$는 평균 θ의 자연스러운 추정량이다. 다음 식들이 성립한다.

$$E\left(\bar{x}_n\right) = \theta, \quad Var\left(\bar{x}_n\right) = \frac{\sigma^2}{n} \tag{3.4.1}$$

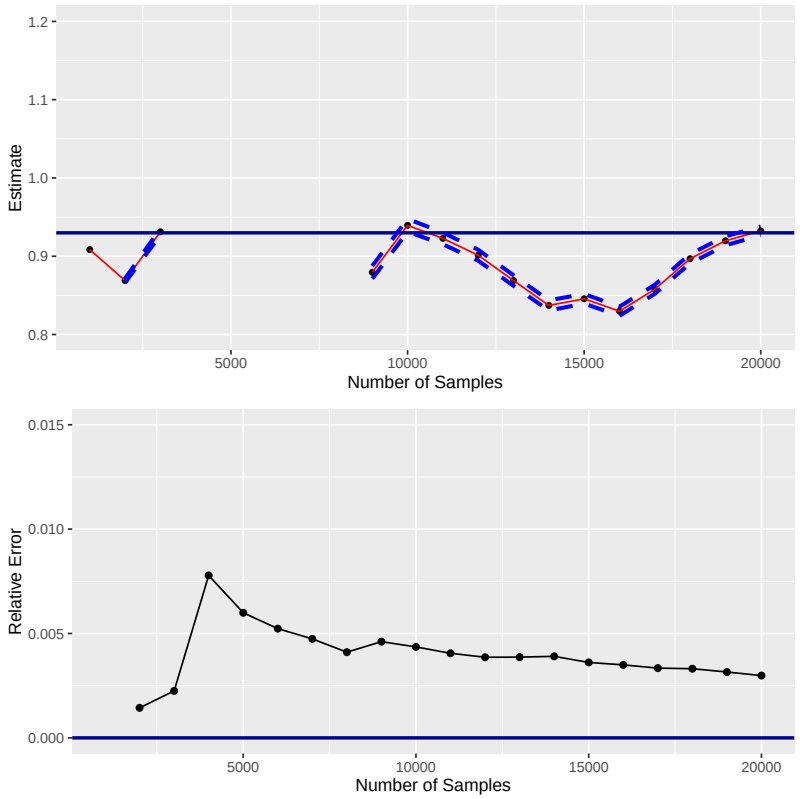


그림 3.3.9. 원시몬테카를로법

3.4.1 대조변량법

확률변수들 x과 y에 대해서, 다음 식이 성립한다.

$$Var\left(\frac{x+y}{2}\right) = \frac{Var\left(x\right) + Var\left(y\right) + 2Cov\left(x,y\right)}{4} \tag{3.4.2}$$

만약 확률변수들 x과 y가 서로 독립이면, 다음 식이 성립한다.

$$Var\left(\frac{x+y}{2}\right) = \frac{Var\left(x\right) + Var\left(y\right)}{4} \tag{3.4.3}$$

만약 확률변수들 x과 y가 음의 상관관계를 이루면, 다음 식이 성립한다.

$$Var\left(\frac{x+y}{2}\right) \leq \frac{Var\left(x\right) + Var\left(y\right)}{4} \tag{3.4.4}$$

즉, 확률변수들 x과 y가 음의 상관관계를 이루면, 이 확률변수들이 독립인 경우에 비해서 분산이 작아진다. 따라서, 음의 상관을 갖는 확률변수들끼리 쌍을 이루게 하면, 표본평균의 분산이 작아지리라 생각된다. 이러한 아이디어를 바탕으로 하는 분산감소법을 대조변량법

(antithetic-variable method)이라 한다.

예제 3.4.1 지지대가 $(0,1)$인 일양확률변수 u에 대해서 다음 식들이 성립함을 알 수 있다.

$$\begin{aligned}\theta &= E\left(e^u\right) = \int_0^1 e^u du \\ &= e-1 \approx 1.7182818284590 46 \qquad (1)\\ Var\left(e^u\right) &= E\left(e^{2u}\right) - E^2\left(e^u\right) = \int_0^1 e^{2u}du - [e-1]^2 \\ &= \frac{e^2-1}{2} - [e-1]^2 \approx 0.2420 \qquad (2)\\ Cov\left(e^u, e^{1-u}\right) &= E\left(e^u e^{1-u}\right) - E\left(e^u\right)E\left(e^{1-u}\right) \\ &= e - [e-1]^2 \approx -0.2342 \qquad (3)\end{aligned}$$

일양샘플들을 $u_1, u_2, \cdots, u_n$이라 할 때, 원시몬테카를로법에 의한 정적분 θ의 추정값은 다음과 같다.

$$\hat{\theta}_C = \frac{1}{n}\sum_{j=1}^{n} \exp\left(u(j)\right) \qquad (4)$$

다음 식들이 성립한다.

$$E\left(\hat{\theta}_C\right) = E\left(e^u\right) = \theta \qquad (5)$$

$$Var\left(\hat{\theta}_C\right) = \frac{1}{n}Var\left(e^u\right) = \frac{0.2420}{n} \qquad (6)$$

확률변수들 e^u와 e^{1-u}는 음의 상관관계를 갖는다. 따라서, 일양샘플들을 $u_1, u_2, \cdots, u_n$이라 할 때, 대조변량법에 의한 정적분 θ의 추정값은 다음과 같다.

$$\hat{\theta}_A = \frac{1}{n}\sum_{j=1}^{n} \frac{\exp\left(u(j)\right) + \exp\left(1-u(j)\right)}{2} \qquad (7)$$

다음 식들이 성립한다.

$$E\left(\hat{\theta}_A\right) = E\left(\frac{\exp(u)+\exp(1-u)}{2}\right) = \theta \tag{8}$$

$$Var\left(\hat{\theta}_A\right) = \frac{1}{n}Var\left(\frac{\exp(u)+\exp(1-u)}{2}\right)$$
$$= \frac{1}{n}\frac{Var\left(e^u\right)+Var\left(e^{1-u}\right)+2Cov\left(e^u,e^{1-u}\right)}{4} \approx \frac{0.0039}{n} \tag{9}$$

식 (9)의 대조변량법에 의한 분산과 식 (6)의 원시몬테카를로법에 의한 분산의 비, 즉 오차분산비는 $0.0039/0.2420 = 0.0162$이다.

원시몬테카를로법과 대조변량법을 비교하기 위해서, 다음 MATLAB프로그램 CrudeAntithetic101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
% Filename: CrudeAntithetic101.m
% Crude vs Antithetic Monte Carlo Integration 1
% Programmed by CBS
% ------------------------------------------------------------------------
clear all, close all
rand('twister',5489);   % Use twister or state for seed.
% Monte Carlo integration
dis = 100;
u = rand(dis*30,1);
NS = dis*(1:30)';
sNS = sqrt(NS);
for ndum = 1:1:30
    dumMC = exp(u(1:ndum*dis));
    CrudeMC(ndum) = mean(dumMC);
    CrudeStd(ndum) = std(dumMC)/sqrt(sNS(ndum));
    CrudeUpper(ndum) = CrudeMC(ndum) + 2*CrudeStd(ndum);
    CrudeLower(ndum) = CrudeMC(ndum) - 2*CrudeStd(ndum);
    dumMC2 = exp(u(1:ndum*dis))+exp(1-u(1:ndum*dis));
    AntitheticMC(ndum) = mean(dumMC2/2);
    AntitheticStd(ndum) = std(dumMC2/2)/sqrt(sNS(ndum));
    AntitheticUpper(ndum) = AntitheticMC(ndum) ...
                          + 2*AntitheticStd(ndum);
    AntitheticLower(ndum) = AntitheticMC(ndum) ...
                          - 2*AntitheticStd(ndum);
    VarRatio(ndum) = AntitheticStd(ndum)^2/CrudeStd(ndum)^2;
end
% Ploting
theta = (exp(1)-1)*ones(30,1);
plot(NS,theta,'k:',NS,CrudeMC,'r-',NS,AntitheticMC,'g--','LineWidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[1.3 2.2])
legend('True Value','Crude','Antithetic','location','NE')
hold on
plot(NS,CrudeUpper,'r:',NS,CrudeLower,'r:','LineWidth',2)
plot(NS,AntitheticUpper,'g:',NS,AntitheticLower,'g:','LineWidth',2)
xlabel('\bf Number of Samples','Fontsize',12');
ylabel('\bf MC Integral','Fontsize',12');
hold off
VarRatio                    % Ratio of Sample variances
saveas(gcf,'CrudeAntithetic101','epsc')
```

```
save('CrudeAntithetic101','CrudeMC','AntitheticMC')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 식 (1)의 정적분을 원시몬테카를로법과 대조변량법으로 추정한다. 일양샘플들의 개수를 100부터 100씩 3000까지 증가시켜가며 계산한 원시몬테카를로추정값과 대조변량법에 의한 추정값이 그림 3.4.1에 그려져 있다. 그림 3.4.1에서 적색 실선은 원시몬테카를로법에 의한 추정값을 나타내고 녹색 긴점선은 대조변량법에 의한 추정값을 나타낸다. 이 그림에서 알 수 있듯이, 원시몬테카를로법에 의한 추정값보다 대조변량법에 의한 추정값이 더 빨리 진짜 정적분으로 수렴하고 또한 신뢰구간의 길이도 훨씬 짧다. 또한, 벡터 VarRatio에서 알 수 있듯이 두 추정값들의 표본분산들의 비는 오차분산비 0.0162로 수렴한다. ■

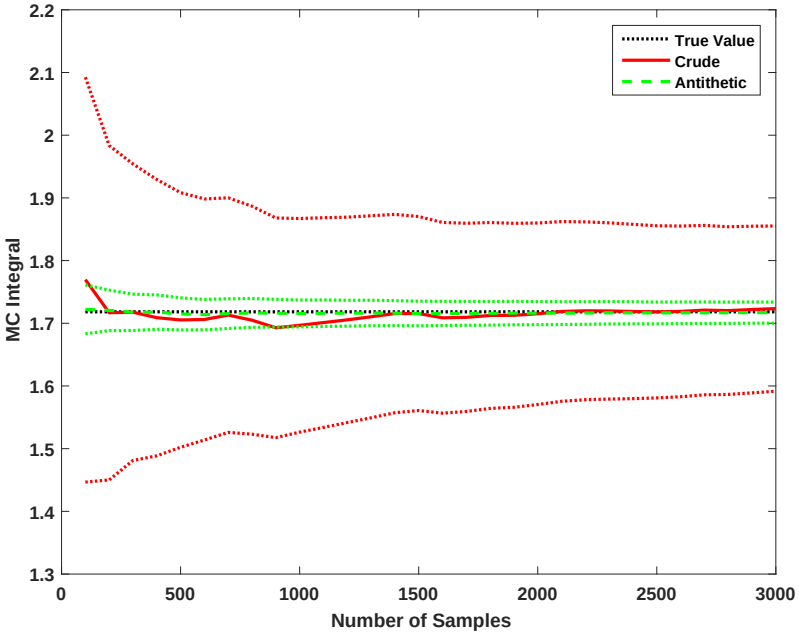


그림 3.4.1. 원시몬테카를로법과 대조변량법 1 (MATLAB)

예제 3.4.2 R언어를 사용해서 예제 3.4.1의 문제를 풀기 위해, 다음 R프로그램 CrudeAntithetic101.R을 실행해 보자.

원시몬테카를로법과 대조변량법을 비교하기 위해서, 다음 R프로그램 CrudeAntithetic101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: CrudeAntithetic101R.R
```

```
# Crude vs Antithetic Monte Carlo Integration 1
# Programmed by CBS
# ------------------------------------------------------------------------
# Generating Random Numbers and Simulations
set.seed(11)
dis <- 100
u <- runif(dis*301)
NS <- dis*c(1:30)
sNS <- sqrt(NS)
NoSim <- 30
CrudeMC <- CrudeSd <- CrudeUpper <- CrudeLower <- rep(0,NoSim)
AntitheticMC <- AntitheticSd <- AntitheticUpper <- rep(0,NoSim)
AntitheticLower <- VarRatio <- rep(0,NoSim)

for( ndum in 1:1:NoSim ){
    dumMC <- exp(u[1:(ndum*dis)])
    CrudeMC[ndum] <- mean(dumMC)
    CrudeSd[ndum] <- sd(dumMC)/sqrt(sNS[ndum])
    CrudeUpper[ndum] <- CrudeMC[ndum] + 2*CrudeSd[ndum]
    CrudeLower[ndum] <- CrudeMC[ndum] - 2*CrudeSd[ndum]
    dumMC2 <- exp(u[1:(ndum*dis)]) + exp(1-u[1:(ndum*dis)])
    AntitheticMC[ndum] <- mean(dumMC2/2)
    AntitheticSd[ndum] <- sd(dumMC2/2)/sqrt(sNS[ndum])
    AntitheticUpper[ndum] <- AntitheticMC[ndum] + 2*AntitheticSd[ndum];
    AntitheticLower[ndum] <- AntitheticMC[ndum] - 2*AntitheticSd[ndum];
    VarRatio[ndum] <- AntitheticSd[ndum]^2/CrudeSd[ndum]^2;
}
VarRatio

# Plotting
# install.packages("ggplot2")
library(ggplot2)
setEPS()
plot.new()
thetaa <- (exp(1)-1)*rep(1,NoSim)
postscript('CrudeAntithetic101R.eps')  # Start to save figure
RVdata <- data.frame(NS,CrudeMC,CrudeUpper,CrudeLower,
                     AntitheticMC,AntitheticUpper,AntitheticLower)
ggplot(RVdata,aes(x=NS,y=CrudeMC)) +
 geom_line(col="red",lwd=1) +
 geom_point(shape=16,col="black",lwd=1.7)+
 geom_hline(yintercept=thetaa,col="dark blue",lwd=1) +
 geom_line(aes(x=NS,y=CrudeUpper),col="red",lwd=1.2,linetype=2) +
 geom_line(aes(x=NS,y=CrudeLower),col="red",lwd=1.2,linetype=2) +
 ylim(1.3,2.2) +
 xlab("Number of Samples") + ylab("MC integral") +
 geom_line(aes(x=NS,y=AntitheticMC),col="green",lwd=1) +
 geom_point(shape=16,col="black",lwd=1.7)+
 geom_hline(yintercept=thetaa,col="dark blue",lwd=1) +
 geom_line(aes(x=NS,y=AntitheticUpper),col="green",lwd=1.2,linetype=2) +
 geom_line(aes(x=NS,y=AntitheticLower),col="green",lwd=1.2,linetype=2)
dev.off()            # End to save figure
# ------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 정적분을 원시몬테카를로법과 대조변량법으로 추정한다. 일양샘플들의 개수를 100부터 100씩 3000까지 증가시켜가며 계산한 원시몬테카를로추정값과 대조변량법에 의한 추정값이 그림 3.4.2에 그려져 있다. 그림 3.4.2에서 적색 실선은 원시몬

테카를로법에 의한 추정값을 나타내고 녹색 긴점선은 대조변량법에 의한 추정값을 나타낸다. 이 그림에서 알 수 있듯이, 원시몬테카를로법에 의한 추정값보다 대조변량법에 의한 추정값이 더 빨리 진짜 정적분으로 수렴하고 또한 신뢰구간의 길이도 훨씬 짧다. 또한, 벡터 VarRatio 에서 알 수 있듯이 두 추정값들의 표본분산들의 비는 오차분산비 0.0162로 수렴한다. ■

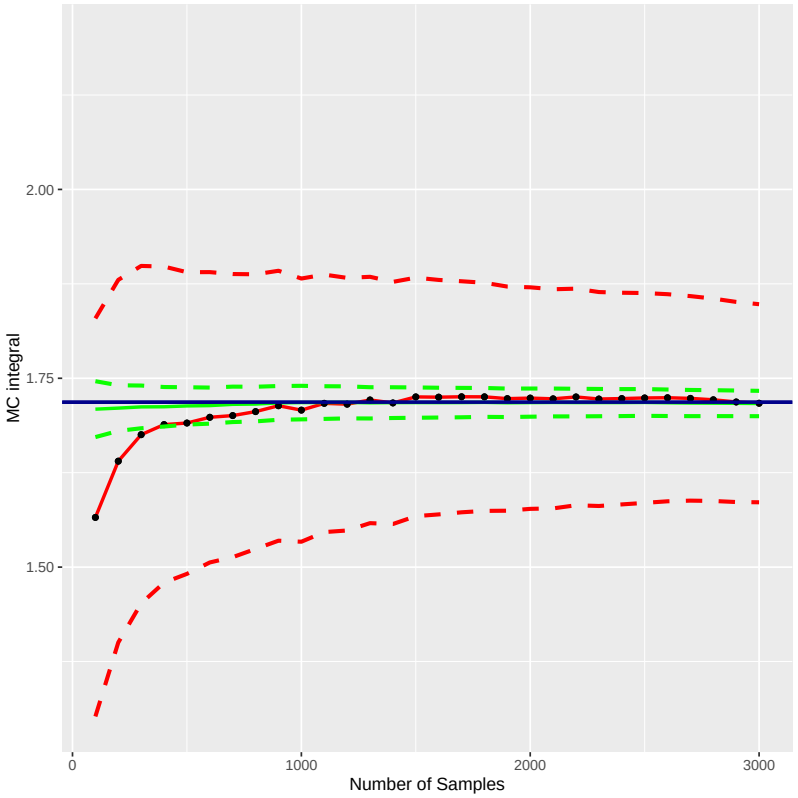


그림 3.4.2. 원시몬테카를로법과 대조변량법 1 (R)

서로 독립이고 지지대가 $(0,1)$인 일양확률변수들 $u_1, u_2, \cdots, u_m$과 함수 h에 대해서, 확률변수 $x = h(u_1, u_2, \cdots, u_m)$이 확률밀도함수 $f(x)$를 갖는다고 하자. 확률변수들 $1-u_1, 1-u_2, \cdots, 1-u_m$도 서로 독립이며 지지대가 $(0,1)$인 일양확률분포를 따르므로, 확률변수 $y = h(1-u_1, 1-u_2, \cdots, 1-u_m)$또한 확률밀도함수 $f(x)$를 갖는다. 만약 함수 h가 단조이면, 확률변수들 x와 y는 음의 상관관계를 갖는다. 따라서, 식 (3.4.4)에서 알 수 있듯이, 대조변량법은 추정량의 분산을 감소시킨다. 확률분포함수 F와 지지대가 $(0,1)$인 일양확률변수들 $u_1, u_2, \cdots, u_n$에 대해서 $\{x(i) = F^{-1}(u(i)) \mid i = 1, 2, \cdots, n\}$는 확률분포함수 F를 따르고 서로 독립인 확률변수들이다. 또한, $\{y(i) = F^{-1}(1-u(i)) \mid i = 1, 2, \cdots, n\}$도 확률분포함수 F를 따르고 서로 독립인 확률변수들이다. 이러한 $\{y(i)\}$를 대조변량열 (sequence of antithetic variables)라 부른다. 확률분포함수 F는 단조증가이므로 역함수 F^{-1}

도 단조증가이다. 따라서, $x(i) = F^{-1}(u(i))$와 $y(i) = F^{-1}(1-u(i))$는 음의 상관관계를 가질 것이다.

정규확률분포인 경우에는 대조변량들을 쉽게 찾을 수 있다. 확률변수 x가 정규확률분포 $\mathcal{N}(\mu, \sigma^2)$를 따르면, 확률변수 $y = 2\mu - x$ 역시 정규확률분포 $\mathcal{N}(\mu, \sigma^2)$를 따른다. 어떤 함수 h에 대해서 $\theta = E(h(x))$를 추정하기로 하자. 확률분포 $\mathcal{N}(\mu, \sigma^2)$를 따르고 서로 독립인 확률변수들 $x_1, x_2, \cdots, x_n$에 대해서, 원시몬테카를로법에 의한 추정량은 다음과 같다.

$$\hat{\theta}_C = \frac{1}{n}\sum_{j=1}^{n} h(x(j)) \tag{3.4.5}$$

다음 식들이 성립한다.

$$E\left(\hat{\theta}_C\right) = \theta \tag{3.4.6}$$

$$Var\left(\hat{\theta}_C\right) = \frac{1}{n}\sigma_h^2 \tag{3.4.7}$$

여기서 $\sigma_h^2 \doteq Var(h(x))$이다. 대조변량법에 의한 추정값은 다음과 같다.

$$\hat{\theta}_A = \frac{1}{n}\sum_{j=1}^{n} \frac{h(x(j)) + h(y(j))}{2} \tag{3.4.8}$$

여기서 $y(j) = 2\mu - x(j)$이다. 다음 식들이 성립한다.

$$E\left(\hat{\theta}_A\right) = \theta \tag{3.4.9}$$

$$Var\left(\hat{\theta}_A\right) = \frac{1}{4n}Var(h(x) + h(y)) = \frac{\sigma_h^2}{n}\frac{[1+\rho_h]}{2} \tag{3.4.10}$$

여기서 $\rho_h \doteq Corr(h(x), h(y))$이다. 식 (3.4.7), 식 (3.4.10), 그리고 부등식 $|\rho_h| \leq 1$에서 알 수 있듯이, 다음 식이 성립한다.

$$Var\left(\hat{\theta}_A\right) \leq Var\left(\hat{\theta}_C\right) \tag{3.4.11}$$

만약 $h(x) = x$이면, 다음 식들이 성립한다.

$$Cov(h(x), h(y)) = Cov(x, 2\mu - x) = -Var(x) \tag{3.4.12}$$

즉, 식 $\rho_h = -1$이 성립하고, 따라서 $\hat{\theta}_A$의 분산은 0이다. 식 $x + y = 2\mu$이 성립하므로, 이는 자명한 결과이다. 따라서, h가 선형함수에 가까울수록, 대조변량의 효과가 크다는 것을 알 수 있다.

예제 3.4.3 표준정규확률변수 z에 대해서 다음 식들이 성립함을 알 수 있다.

$$\theta = E\left(z^3 e^z\right) = E\left(z^3 e^{-z}\right) = \int_{-\infty}^{\infty} z^3 e^z n(z) dz \simeq 6.594885082800514 \tag{1}$$

$$E\left(z^3 e^{-z}\right) = E\left(z^3 e^z\right) = -\theta \tag{2}$$

$$Var\left(z^3 e^z\right) = Var\left(z^3 e^{-z}\right) = E\left(z^6 e^{2z}\right) - \theta^2 = 3687.138993366395 - \theta^2 \tag{3}$$

$$Cov\left(z^3 e^z, z^3 e^{-z}\right) = E\left(z^6\right) - \theta[-\theta] = E\left(z^6\right) + \theta^2 = 15 + \theta^2 \tag{4}$$

원시몬테카를로법에 의한 정적분 θ의 추정값은 다음과 같다.

$$\hat{\theta}_C = \frac{1}{n} \sum_{j=1}^{n} z(j)^3 \exp\left(z(j)\right) \tag{5}$$

다음 식이 성립한다.

$$E\left(\hat{\theta}_C\right) = E\left(z^3 e^z\right) = \theta \tag{6}$$

또한 다음 식이 성립한다.

$$Var\left(\hat{\theta}_C^2\right) = \frac{1}{n} Var\left(z^3 e^z\right) = \frac{1}{n}\left[3687.138993366395 - \theta^2\right] = \frac{1}{n} 60.3626^2 \tag{7}$$

정규샘플들을 $z_1, z_2, \cdots, z_n$이라 할 때, 대조변량법에 의한 정적분 θ의 추정값은 다음과 같다.

$$\hat{\theta}_A = \frac{1}{n} \sum_{j=1}^{n} \frac{z(j)^3 \exp\left(z(j)\right) - z(j)^3 \exp\left(-z(j)\right)}{2} = \frac{1}{n} \sum_{j=1}^{n} z(j)^3 \frac{\exp\left(z(j)\right) - \exp\left(-z(j)\right)}{2} \tag{8}$$

다음 식들이 성립한다.

$$E\left(\hat{\theta}_A\right) = E\left(z^3 \frac{e^z - e^{-z}}{2}\right) = \theta \tag{9}$$

$$\begin{aligned} Var\left(\hat{\theta}_A\right) &= \frac{1}{n} Var\left(z^3 \frac{e^z - e^{-z}}{2}\right) \\ &= \frac{1}{n} \frac{Var\left(z^3 e^z\right) + Var\left(z^3 e^{-z}\right) - 2Cov\left(z^3 e^z, z^3 e^{-z}\right)}{4} \\ &= \frac{1}{n} \frac{Var\left(z^3 e^z\right) - Cov\left(z^3 e^z, z^3 e^{-z}\right)}{2} \\ &= \frac{1}{n} \frac{E\left(z^6 e^{2z}\right) - 2\theta^2 - 15}{2} = \frac{42.3388^2}{n} \end{aligned} \tag{10}$$

여기서 두 번째 등호는 식 (3)과 식 (4)에 의해서 성립한다.

식 (10)의 대조변량법에 의한 분산과 식 (7)의 원시몬테카를로법에 의한 분산의 비, 즉 오차분산비는 $42.3388^2/60.3626^2 = 0.4920$이다.

원시몬테카를로법과 대조변량법을 비교하기 위해서, 다음 MATLAB프로그램 CrudeAntithetic102.m을 실행해 보자.

```
% ------------------------------------------------------------------------
% Filename: CrudeAntithetic102.m
% Crude vs Antithetic Monte Sarlo Integration 2
% Programmed by SBS
% ------------------------------------------------------------------------
clear all, close all
diary CrudeAntithetic102.txt
syms x
theta = int(x.^3.*exp(x).*1./sqrt(2*pi).*exp(-1/2*x.^2),x,-inf,inf)
theta = double(theta)
% Monte Sarlo integration
rand('twister',5489);   % Use twister or state for seed.
z = randn(1);
Sc(1) = z^3*exp(z);
Sa(1) = z^3*(exp(z)-exp(-z))/2;
Vc(1) = 0; Va(1) = 0; Vratio(1) = 1;
d = 0.05;
for ndum=2:10000000
    z = randn(1);
    dumx = z^3*exp(z);
    duma = z^3*(exp(z)-exp(-z))/2;
    Sc(ndum) = (1-1/ndum)*Sc(ndum-1) + 1/ndum*dumx;
    Sa(ndum) = (1-1/ndum)*Sa(ndum-1) + 1/ndum*duma;
    Vc(ndum) = (1-1/(ndum-1))*Vc(ndum-1)+(Sc(ndum-1)-Sc(ndum))^2 ...
               + (dumx-Sc(ndum))^2/(ndum-1);
    Va(ndum) = (1-1/(ndum-1))*Va(ndum-1)+(Sa(ndum-1)-Sa(ndum))^2 ...
               + (duma-Sa(ndum))^2/(ndum-1);
    Vratio(ndum)= Va(ndum)/Vc(ndum);
    if sqrt(Vc(ndum)/ndum) < d
       break
    end
end
ndum, Sc(end), Sa(end), Vratio(end)
```

```
% Plotting
NSdum = floor(ndum/40)
NS = NSdum*(1:40);
thetaT = theta*ones(1,40);
CrudeMC = Sc(NS);
AntitheticMC = Sa(NS);
VratioMC = Vratio(NS);
subplot(2,1,1)
plot(NS,thetaT,'k:',NS,CrudeMC,'r-',NS,AntitheticMC,'g--','LineWidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0,NS(end)])
legend('True Value','Crude','Antithetic','location','SE')
xlabel('Number of Random Numbers')
ylabel('MC estimates')
subplot(2,1,2)
plot(NS,VratioMC,'k-','LineWidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0,NS(end)])
ylabel('Variance Ratio')
saveas(gcf,'CrudeAntithetic102','epsc')
save('CrudeAntithetic102','CrudeMC','AntitheticMC')
diary off
% End of program
% --------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 식 (1)의 정적분 θ를 원시몬테카를로법과 대조변량법으로 추정한다. 대조변량법에 의한 추정량의 표준오차가 $d = 0.05$미만이 될 때까지 일양샘플들의 개수를 1부터 1개씩 증가시켜가며 추정값과 추정값의 표본분산을 계산한다. 이러한 계산을 효율적으로 하기 위해서 다음 점화식들을 사용한다. 우선, 표본평균 $\bar{x}_n$을 처음 $n-1$개의 관찰값들 $x_1, x_2, \cdots, x_{n-1}$과 마지막 관찰값 x_n을 분할해서 계산하면, 다음 식을 얻는다.

$$\bar{x}_n = \left[1 - \frac{1}{n}\right] \bar{x}_{n-1} + \frac{1}{n} x_n \tag{11}$$

점화식 (1)의 초기값은 $\bar{x}_1 = x_1$이다. 식 (1)을 사용하면, 제 n번째 관찰값을 얻었을 때 n개의 관찰값들을 모두 사용해서 표본평균을 계산할 필요는 없다. 그러나, 식 (1)은 점화식이므로, 오차의 축적에 유의해야 한다. 다음 식이 성립함을 알 수 있다.

$$s_n^2 = \left[1 - \frac{1}{n-1}\right] s_{n-1}^2 + [\bar{x}_{n-1} - \bar{x}_n]^2 + \frac{1}{n-1} [x_n - \bar{x}_n]^2 \tag{12}$$

점화식 (2)의 초기값은 $s_2^2 = [x_1 - x_2]^2 / 2$이다. 식 (2)도 점화식이므로, 오차의 축적에 유의해야 한다. 식 (2)를 사용해서, 표준오차 $s_n/\sqrt{n}$을 순차적으로 계산할 수 있다. 식 (1)과 식 (2)에 대해서는 최병선 [**?**, 제4.2.7소절]을 참조하라.

이 MATLAB프로그램을 실행하면, $n = 1,845,248$에서 표준오차가 처음으로 $d = 0.05$미만이 된다. 이때, 원시몬테카를로법에 의한 정적분 θ의 추정값은 $\hat{\theta}_C = 6.6515$이고, 대조변

량법에 의한 정적분 θ의 추정값은 $\hat{\theta}_C = 6.6597$이다. 원시몬테카를로추정값과 대조변량법에 의한 추정값이 그림 3.4.3의 위 그래프에 그려져 있다. 이 그래프에서 적색 실선은 원시몬테카를로법에 의한 추정값을 나타내고 녹색 긴점선은 대조변량법에 의한 추정값을 나타낸다. 그림 3.4.3의 하단 그래프는 오차분산비를 그린 것이다. 이 그래프에서 알 수 있듯이 두 추정값들의 표본분산들의 비는 0.5 근처로 수렴한다. ■

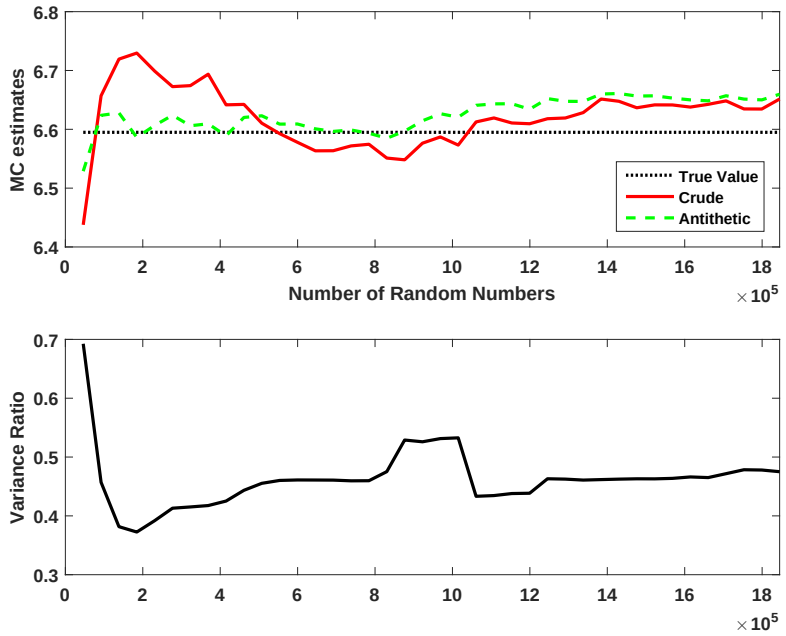


그림 3.4.3. 원시몬테카를로법과 대조변량법 2 (MATLAB)

예제 3.4.4 R언어를 사용해서 예제 3.4.3의 문제를 풀기 위해, 다음 R프로그램 CrudeAntithetic102R.R을 실행해 보자.

```
# ------------------------------------------------------------------------------
# Filename: CrudeAntithetics102R.R
# Crude vs Antithetic Monte Carlo
# Programmed by CBS
# ------------------------------------------------------------------------------
# True Value
thetaa <- 6.5948850828
# Generating Random Numbers and Simulations
set.seed(11)
nSample <- 10000000
Sc <- Sa <- Vc <- Va <- Vratio <- rep(0,nSample)
z <-rnorm(1)
Sc[1] <- z^3*exp(z)
Sa[1] <- z^3*(exp(z)-exp(-z))/2
Vc[1] <- Va[1] <- 0
Vratio[1] <- 1
d <- 0.05
```

```
for ( ndum in 2:nSample){
    z <- rnorm(1)
    dumx <- z^3*exp(z)
    duma <- z^3*(exp(z)-exp(-z))/2
    Sc[ndum] = (1-1/ndum)*Sc[ndum-1] + 1/ndum*dumx
    Sa[ndum] = (1-1/ndum)*Sa[ndum-1] + 1/ndum*duma;
    Vc[ndum] = (1-1/(ndum-1))*Vc[ndum-1]+
               (Sc[ndum-1]-Sc[ndum])^2 + (dumx-Sc[ndum])^2/(ndum-1)
    Va[ndum] = (1-1/(ndum-1))*Va[ndum-1]+(Sa[ndum-1]-Sa[ndum])^2 +
               (duma-Sa[ndum])^2/(ndum-1)
    Vratio[ndum]= Va[ndum]/Vc[ndum]
    if ( Vc[ndum]/ndum < d^2 ) break
}
ndum; Sc[ndum]; Sa[ndum]; Vratio[ndum]

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('CrudeAntithetic102R.eps')  # Start to save figure
NSdum <- floor(ndum/40)
NS <- NSdum*c(1:40)
thetaT <- thetaa*rep(1,40)
CrudeMC <- Sc[NS]
AntitheticMC <- Sa[NS]
VratioMC <- Vratio[NS]
RVdata <- data.frame(thetaT,CrudeMC,AntitheticMC,VratioMC)
plotMC <- ggplot(RVdata,aes(x=NS,y=CrudeMC)) +
          geom_point(shape=16,col="red",lwd=1.7) +
          geom_line(col="red",linetype=1) +
          geom_point(aes(x=NS,y=AntitheticMC),shape=16,col="green",lwd=1.7) +
          geom_line(aes(x=NS,y=AntitheticMC),col="green",linetype=2) +
          geom_hline(yintercept=thetaa,col="dark blue",lwd=1) +
          ylim(5.5,7) +
          xlab("Number of Samples") + ylab("MC estimate")
plotVR <- ggplot(RVdata,aes(x=NS,y=VratioMC)) +
          geom_point(shape=16,col="black",lwd=1.7) +
          geom_line(col="black",linetype=1) +
          ylim(0,1.0) +
          xlab("Number of Samples") + ylab("Variance Ratio")
pushViewport(viewport(layout=grid.layout(2,1)))
print(plotMC, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plotVR, vp=viewport(layout.pos.row=2, layout.pos.col=1))
dev.off()            # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, θ를 원시몬테카를로법과 대조변량법으로 추정한다. 대조변량법에 의한 추정량의 표준오차가 $d = 0.05$미만이 될 때까지 일양샘플들의 개수를 1부터 1개씩 증가시켜가며 추정값과 추정값의 표본분산을 계산한다. 이러한 계산을 효율적으로 하기 위해서 예제 3.4.3의 점화식들을 사용한다.

이 R프로그램을 실행하면, $n = 1,509,475$에서 표준오차가 처음으로 $d = 0.05$미만이 된다. 이때, 원시몬테카를로법에 의한 정적분 θ의 추정값은 $\hat{\theta}_C = 6.619827$이고, 대조변량법에

의한 정적분 θ의 추정값은 $\hat{\theta}_C = 6.604948$이다. 원시몬테카를로추정값과 대조변량법에 의한 추정값이 그림 3.4.4의 위 그래프에 그려져 있다. 이 그래프에서 적색 실선은 원시몬테카를로법에 의한 추정값을 나타내고 녹색 긴점선은 대조변량법에 의한 추정값을 나타낸다. 그림 3.4.4의 하단 그래프는 오차분산비를 그린 것이다. 이 그래프에서 알 수 있듯이 두 추정값들의 표본분산들의 비는 0.5 근처로 수렴한다. ■

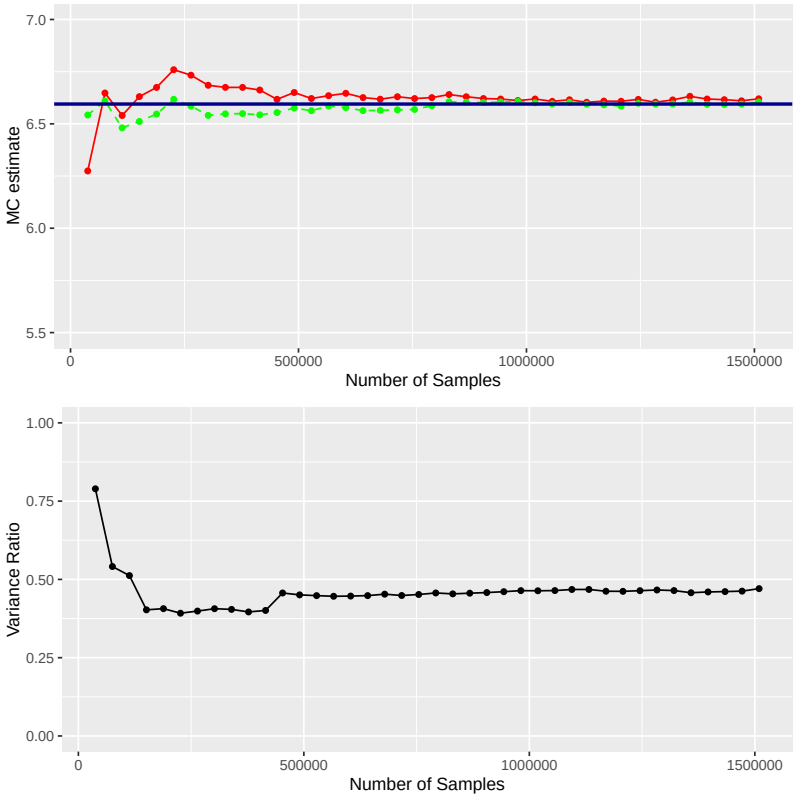


그림 3.4.4. 원시몬테카를로법과 대조변량법 2 (R)

3.4.2 제어변량법

확률변수 x의 기대값 $\theta = E(x)$를 추정하기 위해서, 기대값 μ를 알고 있는 확률변수 z를 이용하기로 하자. 임의의 상수 c에 대해서 확률변수 $x - c[z - \mu]$도 기대값 θ의 불편추정량이다. 즉, 제어변량(control variable) z를 사용해서 확률변수 x의 기대값 θ를 추정한다. 확률변수 $x - c[z - \mu]$의 분산은 다음과 같다.

$$Var(x - c[z - \mu]) = Var(x - cz) = Var(x) - 2cCov(x, z) + c^2 Var(z) \tag{3.4.13}$$

식 (3.4.13)에서 알 수 있듯이, $Var(x-c[z-\mu])$는 c의 2차함수이다. 또한, $Var(x-c[z-\mu])$를 최소화하는 상수 c는 다음과 같다.

$$c^* \doteq \frac{Cov(x,z)}{Var(z)} \tag{3.4.14}$$

다음 식이 성립한다.

$$Var(x-c^*[z-\mu]) = Var(x) - \frac{1}{Var(z)}Cov(x,z)^2 \tag{3.4.15}$$

따라서, 다음 식이 성립한다.

$$\frac{1}{Var(x)}Var(x-c^*[z-\mu]) = 1-\rho_{x,z}^2 \tag{3.4.16}$$

여기서 $\rho_{x,z}$는 확률변수들 x와 z의 상관계수이다. 이 상관계수가 크면, 분산이 크게 감소함을 알 수 있다. 이것은 직관적으로 보아 당연한 것이다. 종속변수를 x로 하고 설명변수를 z로 하는 선형회귀분석에서 $x-c^*[z-\mu]$는 잔차에 해당한다. 따라서, 설명변수와 종속변수의 상관계수가 크면, 이 잔차들의 절대값이 작고 결과적으로 분산도 작다.

예제 3.4.5 예제 3.4.1에서 다룬 $\theta = \int_0^1 e^u du$를 추정하는 문제를 다시 살펴보자. 제어변량법으로 θ를 추정하기 위해서 u를 제어변량으로 사용하기로 하자. 다음 식들이 성립한다.

$$\begin{aligned} &Cov\left(e^u, u\right) = E\left(ue^u\right) - E\left(u\right)E\left(e^u\right) \\ &= \int_0^1 ue^u du - \frac{1}{2}[e-1] = \frac{1}{2}[3-e] \approx 0.140859085770477 \end{aligned} \tag{1}$$

따라서, 다음 식들이 성립한다.

$$c^* = \frac{Cov\left(e^u, u\right)}{Var\left(u\right)} = \frac{\left[3-\exp(1)\right]/2}{1/12} \approx 1.6903 \tag{2}$$

식 (3.4.15)에서 알 수 있듯이, 다음 식들이 성립한다.

$$\begin{aligned} &Var\left(e^u - c^*\left[u-\frac{1}{2}\right]\right) = Var\left(e^u\right) - \frac{1}{Var(u)}Cov\left(e^u,u\right)^2 \\ &= \frac{4e-3-e^2}{2} - 12\cdot\left[\frac{3-e}{2}\right]^2 \approx 0.0039 \end{aligned} \tag{3}$$

예제 3.4.1에서 알 수 있듯이, 일양샘플들이 $u_1, u_2, \cdots, u_n$ 인 경우, 원시몬테카를로법에 의한 추정량은 다음과 같다.

$$\hat{\theta}_C = \frac{1}{n}\sum_{j=1}^{n} \exp\left(u(j)\right) \tag{4}$$

또한, 다음 식들이 성립한다.

$$E\left(\hat{\theta}_C\right) = E\left(e^u\right) = \theta, Var\left(\hat{\theta}_C\right) = \frac{1}{n}, \;\; Var\left(e^u\right) = \frac{0.2420}{n} \tag{5}$$

이 제어변량법에 의한 추정값은 다음과 같다.

$$\hat{\theta}_{CV} = \frac{1}{n}\sum_{j=1}^{n} \left\{\exp\left(u(j)\right) - c^* \left[u(j) - \frac{1}{2}\right]\right\} \tag{6}$$

식 (5) 에서 알 수 있듯이, 다음 식들이 성립한다.

$$E\left(\hat{\theta}_{CV}\right) = E\left(\exp\left(u\right) - c^* \left[u - \frac{1}{2}\right]\right) = \theta \tag{7}$$

식 (3) 에서 알 수 있듯이, 다음 식들이 성립한다.

$$Var\left(\hat{\theta}_{CV}\right) = \frac{1}{n} Var\left(\exp\left(u\right) - c^* \left[u - \frac{1}{2}\right]\right) \approx \frac{0.0039}{n} \tag{8}$$

식 (8) 의 제어변량법에 의한 분산과 식 (5) 의 원시몬테카를로법에 의한 분산의 비, 즉 오차분산비는 $0.0039/0.2420 = 0.0162$ 이다.

이 예제에서는 $\theta = \int_0^1 e^u du$ 를 추정하기 위해서, 식 (2) 의 c^* 를 먼저 구해야 한다. 그러나, c^* 를 구하는 것은 θ 를 구하는 것 못지 않게 복잡하다. 따라서, 이 예제에서 제어변량법은 현실적인 것이 아니다. 임의의 c^* 에 대해서 식 (7) 이 성립하므로, c^* 의 근사값 $\hat{c}$ 를 사용해도 된다. 다만, 근사값 $\hat{c}$ 를 사용하면 c^* 를 사용할 때보다 분산이 커질 따름이다. 그러나, $\sum_{j=1}^{n} \hat{c}\left[u(j) - \frac{1}{2}\right]$ 의 기대값은 0이지만 추정값은 0이 아니므로, 이 차이에서 발생하는 오차는 근사값 $\hat{c}$ 의 절대값이 크기에 비례한다.

원시몬테카를로법과 제어변량법을 비교하기 위해서, 다음 MATLAB프로그램 CrudeControl101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
% Filename: CrudeControl101.m
```

```
% Crude vs Control Variate Monte Carlo Integration 1
% Programmed by CBS
% ------------------------------------------------------------------------
clear all, close all
rand('twister',5489);   % Use twister or state for seed.
% Monte Carlo integration
Ns = 3000                    % Number of Samples
u = rand(Ns,1);
for cdum = 1:1:40
    chat(cdum) = cdum/10-0.9;
    dumMC = exp(u);
    CrudeMC(cdum) = mean(dumMC);
    CrudeStd(cdum) = std(dumMC)/sqrt(Ns);
    CrudeUpper(cdum) = CrudeMC(cdum) + 2*CrudeStd(cdum);
    CrudeLower(cdum) = CrudeMC(cdum) - 2*CrudeStd(cdum);
    dumMC2 = dumMC - chat(cdum)*(u-0.5);
    ControlMC(cdum) = mean(dumMC2);
    ControlStd(cdum) = std(dumMC2)/sqrt(Ns);
    ControlUpper(cdum) = ControlMC(cdum) + 2*ControlStd(cdum);
    ControlLower(cdum) = ControlMC(cdum) - 2*ControlStd(cdum);
    VarRatio(cdum) = ControlStd(cdum)^2/CrudeStd(cdum)^2;
end
% Ploting
% MC Integral
theta = exp(1)-1
thet = theta*ones(1,40);
cstar = 6*(3-exp(1))
subplot(1,2,1)
plot(chat,thet,'b:',chat,CrudeMC,'r-',chat,ControlMC,'g--','LineWidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
legend('True Value','Crude','Control Variate','location','NE')
hold on
plot(chat,CrudeUpper,'r:',chat,CrudeLower,'r:','LineWidth',1.5)
plot(chat,ControlUpper,'g:',chat,ControlLower,'g:','LineWidth',1.5)
plot([theta,theta],[1.675 1.7184],'b:','LineWidth',1.5)
xlabel('\it c','Fontsize',12','Fontweigh','bold');
ylabel('MC Integral','Fontsize',12','Fontweigh','bold');
text(cstar-0.15,1.678,'\bf c^{*}')
axis([-0.8 3.0 1.68 1.76])
hold off
% Ratio of Sample Variances
subplot(1,2,2)
plot(chat,VarRatio,'k-',chat,ones(1,40),'b-','LineWidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[-0.8 3.0])
xlabel('\it c','Fontsize',12','Fontweigh','bold');
ylabel('Variance Ratio','Fontsize',12','Fontweigh','bold');
text(cstar-0.15, -0.08,'\bf c^{*}')
saveas(gcf,'CrudeControl101','epsc')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, θ를 원시몬테카를로법과 제어변량법으로 추정한다. 일양샘플들의 개수 3000으로 하고, 원시몬테카를로추정값과 c를 -0.4에서 3.1까지 증가시켜가며 계산한 추정값 $\hat{\theta}(c) \doteq \frac{1}{n}\sum_{j=1}^{n}\left\{\exp\left(u(j)\right) - c\left[u(j) - \frac{1}{2}\right]\right\}$이 그림 **??**에 그려져 있다. 그림 3.4.5의 상단그래프에서 적색 실선은 원시몬테카를로법에 의한 추정값 $\hat{\theta}_C$를 나타내고 녹색 긴점선은 제어변량법에 의한 추정값 $\hat{\theta}_{CV}(c)$를 나타낸다. 이 그림에서 알 수 있듯이,

적당한 c에 대해서는 원시몬테카를로법에 의한 추정값보다 제어변량법에 의한 추정값이 진짜 정적분에 가깝고 또한 신뢰구간의 길이도 훨씬 짧다. 그러나, $c^* = 1.6903$에서 멀리 떨어진 c에 대해서는 그 반대 현상이 일어난다.

식 (3.4.13)에서 알 수 있듯이, $Var(x - c[y - \mu])$는 c의 2차함수이다. 따라서, 원시몬테카를로법에 의한 추정값 $\hat{\theta}_C$와 제어변량법에 의한 추정값 $\hat{\theta}(c)$의 표본오차분산비 VarRatio도 c의 2차함수 형태를 갖을 것이다. 이 표본오차분산비가 그림 3.4.5의 하단 그래프에 그려져 있다. 이 그래프에서 알 수 있듯이, c^* 주변에서 표본오차분산비가 가장 작다. ■

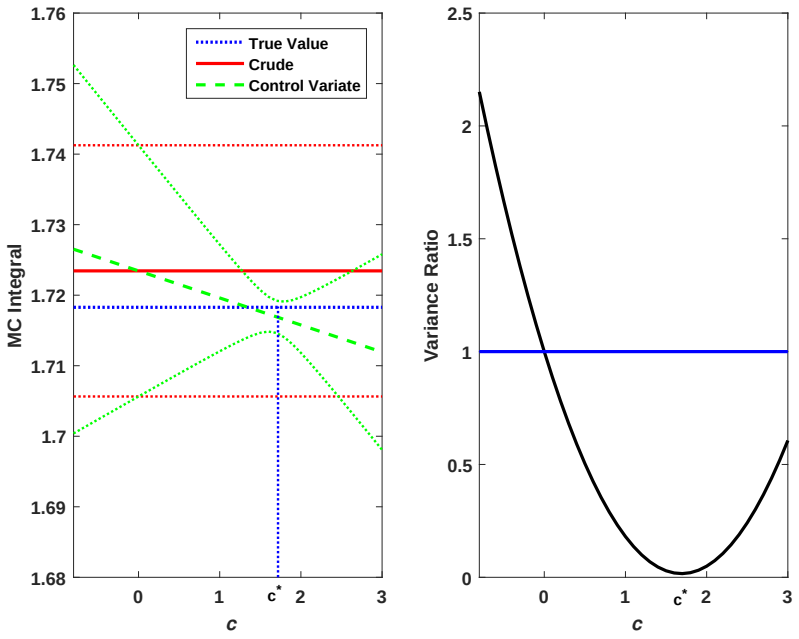


그림 3.4.5. 원시몬테카를로법과 제어변량법 (MATLAB)

예제 3.4.6 R언어를 사용해서 예제 3.4.5의 문제를 풀기 위해서, 다음 R프로그램 CrudeControl101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: CrudeControl101R.R
# Crude vs Control Variate Monte Carlo Integration 1
# Programmed by CBS
# ------------------------------------------------------------------------
# Generating Random Numbers and Simulations
set.seed(5)
Ns <- 3000
u <- runif(Ns)
dumMC <- exp(u)
NoSim <- 40
chat <- rep(0,NoSim)
CrudeMC <- CrudeSd <- CrudeUpper <- CrudeLower <- rep(0,NoSim)
ControlMC <- ControlSd <- ControlUpper <- rep(0,NoSim)
ControlLower <- VarRatio <- rep(0,NoSim)
```

```
16
17 for( cdum in 1:1:NoSim ){
18     chat[cdum] <- cdum/10 - 0.9
19     CrudeMC[cdum] <- mean(dumMC)
20     CrudeSd[cdum] <- sd(dumMC)/sqrt(Ns)
21     CrudeUpper[cdum] <- CrudeMC[cdum] + 2*CrudeSd[cdum]
22     CrudeLower[cdum] <- CrudeMC[cdum] - 2*CrudeSd[cdum]
23     dumMC2 <- dumMC - chat[cdum]*(u-0.5)
24     ControlMC[cdum] <- mean(dumMC2)
25     ControlSd[cdum] <- sd(dumMC2)/sqrt(Ns)
26     ControlUpper[cdum] <- ControlMC[cdum] + 2*ControlSd[cdum]
27     ControlLower[cdum] <- ControlMC[cdum] - 2*ControlSd[cdum]
28     VarRatio[cdum] <- ControlSd[cdum]^2/CrudeSd[cdum]^2
29 }
30
31 # Plotting
32 # Plotting
33 # install.packages("ggplot2")
34 library(ggplot2)
35 # install.packages("grid")
36 library(grid)
37 setEPS()
38 plot.new()
39 thetaa <- exp(1)-1
40 thet <- thetaa*rep(1,NoSim)
41 postscript('CrudeControl101R.eps')  # Start to save figure
42 RVdata <- data.frame(chat,thet,CrudeMC,CrudeUpper,CrudeLower,
43                      ControlMC,ControlUpper,ControlLower)
44 plotMC <- ggplot(RVdata,aes(x=chat,y=CrudeMC)) +
45           geom_line(col="red",lwd=1) +
46           geom_hline(yintercept=thetaa,col="dark blue",lwd=1) +
47           geom_line(aes(x=chat,y=CrudeUpper),col="red",lwd=1.2,linetype=2) +
48           geom_line(aes(x=chat,y=CrudeLower),col="red",lwd=1.2,linetype=2) +
49           xlim(-0.8,3) + ylim(1.68,1.76) +
50           xlab("c") + ylab("MC integral") +
51           geom_vline(xintercept=1.6903,linetype=3,lwd=1,col="dark red") +
52           geom_line(aes(x=chat,y=ControlMC),col="green",lwd=1) +
53           geom_line(aes(x=chat,y=ControlUpper),col="green",lwd=1.2,linetype=2)+
54           geom_line(aes(x=chat,y=ControlLower),col="green",lwd=1.2,linetype=2)
55 plotVR <- ggplot(RVdata,aes(x=chat,y=VarRatio)) +
56           geom_line(col="black",linetype=1,lwd=1.2) +
57           geom_hline(yintercept=1,col="dark blue",lwd=1) +
58           xlim(-0.8,3) +
59           xlab("c") + ylab("Variance Ratio")
60 pushViewport(viewport(layout=grid.layout(2,1)))
61 print(plotMC, vp=viewport(layout.pos.row=1, layout.pos.col=1))
62 print(plotVR, vp=viewport(layout.pos.row=2, layout.pos.col=1))
63 dev.off()             # End to save figure
64 # ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, θ를 원시몬테카를로법과 제어변량법으로 추정한다. 일양샘플들의 개수 3000으로 하고, 원시몬테카를로추정값과 c를 -0.4에서 3.1까지 증가시켜가며 계산한 추정값 $\hat{\theta}(c) \doteq \frac{1}{n}\sum_{j=1}^{n}\left\{\exp\left(u(j)\right) - c\left[u(j) - \frac{1}{2}\right]\right\}$이 그림 3.4.6에 그려져 있다. 그림 3.4.6의 상단그래프에서 적색 실선은 원시몬테카를로법에 의한 추정값 $\hat{\theta}_C$를 나타내고 녹색 긴점선은 제어변량법에 의한 추정값 $\hat{\theta}_{CV}(c)$를 나타낸다. 이 그림에서 알 수 있듯이, 적당한 c

에 대해서는 원시몬테카를로법에 의한 추정값보다 제어변량법에 의한 추정값이 진짜 정적분에 가깝고 또한 신뢰구간의 길이도 훨씬 짧다. 그러나, $c^* = 1.6903$에서 멀리 떨어진 c에 대해서는 그 반대 현상이 일어난다.

식 (3.4.13)에서 알 수 있듯이, $Var(x - c[y - \mu])$는 c의 2차함수이다. 따라서, 원시몬테카를로법에 의한 추정값 $\hat{\theta}_C$와 제어변량법에 의한 추정값 $\hat{\theta}(c)$의 표본오차분산비 VarRatio도 c의 2차함수 형태를 갖을 것이다. 이 표본오차분산비가 그림 3.4.6의 하단 그래프에 그려져 있다. 이 그래프에서 알 수 있듯이, c^* 주변에서 표본오차분산비가 가장 작다. ■

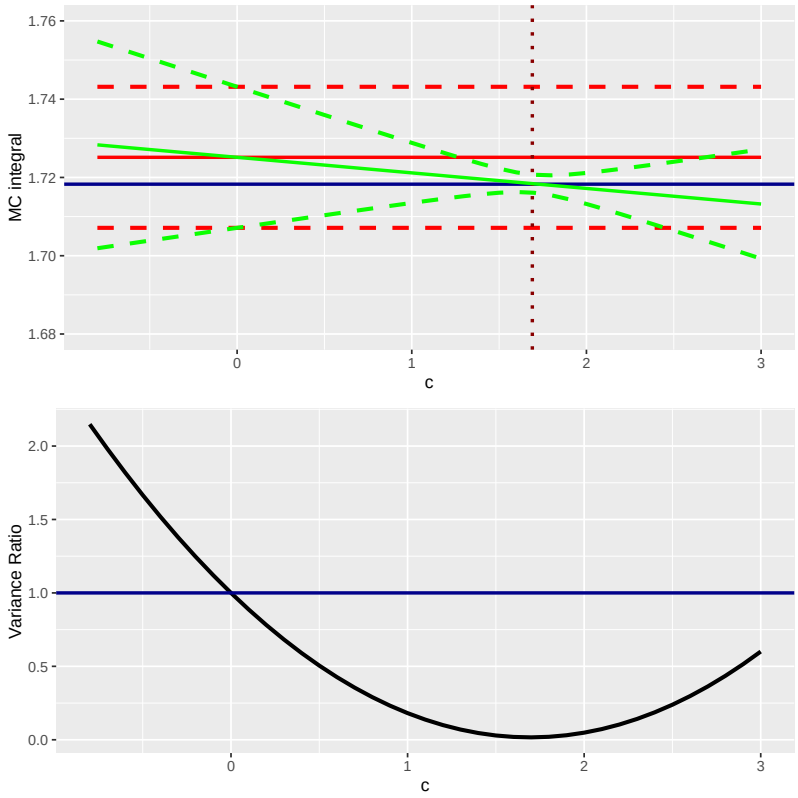


그림 3.4.6. 원시몬테카를로법과 제어변량법 (R)

확률변수 x의 기대값 $\theta = E(x)$를 추정하기 위해서, 기대값들 $\mu_1, \mu_2, \cdots, \mu_m$를 알고 있는 확률변수들 $z_1, z_2, \cdots, z_m$를 이용하기로 하자. 임의의 상수들 $c_1, c_2, \cdots, c_m$에 대해서 확률변수 $v \doteq x - \sum_{i=1}^{m} c(i)[z(i) - \mu(i)]$도 기대값 θ의 불편추정량이다. 즉, 제어변량들 $z_1, z_2, \cdots, z_m$를 이용해서 확률변수 x의 기대값 θ를 추정한다. 이러한 방법을 다변량제어변량법이라 한다. 확률변수 v의 분산은 다음과 같다.

$$Var(v) = Var(x) - 2\sum_{i=1}^{m} c(i)Cov\,(x, z(i)) + \sum_{i=1}^{m}\sum_{j=1}^{m} c(i)c(j)Cov\,(z(i), z(j)) \qquad (3.4.17)$$

선형회귀분석이론을 적용하면, $Var(v)$를 최소화하는 $\boldsymbol{c} = [c_1, c_2, \cdots, c_m]^t$는 다음 연립방정식을 만족한다.

$$C_{zz}\boldsymbol{c} = \boldsymbol{c}_{zx} \tag{3.4.18}$$

여기서 C_{zz}과 $\boldsymbol{c}_{zx}$는 각각 다음과 같다.

$$C_{zz} \doteq \begin{bmatrix} Cov(z_1, z_1) & Cov(z_1, z_2) & \cdots & Cov(z_1, z_m) \\ Cov(z_2, z_1) & Cov(z_2, z_2) & \cdots & Cov(z_2, z_m) \\ \vdots & \vdots & & \vdots \\ Cov(z_m, z_1) & Cov(z_m, z_2) & \cdots & Cov(z_m, z_m) \end{bmatrix}, \quad \boldsymbol{c}_{zx} \doteq \begin{bmatrix} Cov(z_1, x) \\ Cov(z_2, x) \\ \vdots \\ Cov(z_m, x) \end{bmatrix} \tag{3.4.19}$$

이 연립방정식의 해를 $\boldsymbol{c}^* = [c_{1^*}, c_{2^*}, \cdots, c_{m^*}]^t$로 표기하자. 대부분의 상용프로그램에는 이 연립방정식을 푸는 함수를 포함하고 있다 . 확률변수 $v^* \doteq x - \sum_{i=1}^{m} c(i)^*[z(i) - \mu(i)]$의 분산은 다음과 같다.

$$Var(v^*) = Var(x) - \boldsymbol{c}_{zx}^t\boldsymbol{c}^* = Var(x) - \boldsymbol{c}_{\boldsymbol{zx}}^{\boldsymbol{t}} C_{zz}^{-1}\boldsymbol{c}_{\boldsymbol{zx}} \tag{3.4.20}$$

행렬 C_{zz}가 양반정치이므로, 다음 부등식이 성립한다.

$$Var(v^*) \leq Var(x) \tag{3.4.21}$$

예제 3.4.7 예제 3.3.3에서 다룬 π값을 구하는 두 번째 방법을 다시 살펴보자. 이 방법은 다음 식을 이용하는 것이다.

$$\theta \doteq \frac{\pi}{4} = \int_0^1 \sqrt{1 - u^2}du \tag{1}$$

만약 $u_1, u_2, \cdots, u_n$이 지지대가 $(0, 1)$인 일양샘플이면, 원시몬테카를로법에 의한 추정량은 다음과 같다.

$$\hat{\theta}_C = \frac{4}{n}\sum_{j=1}^{n}\sqrt{1 - u(j)^2} \tag{2}$$

이 추정량 $\hat{\theta}_C$의 분산은 다음과 같다.

$$Var\left(\hat{\theta}_C\right) \approx \frac{0.7971}{n} \tag{3}$$

다음 다변량제어변량법에 의한 추정값을 살펴보자.

$$\hat{\theta}_{CV} = \frac{1}{n}\sum_{j=1}^{n}\left(4\sqrt{1-u(j)^2} - c_1^*\left[\frac{1}{2} - u(j)\right] - c_2^*\left\{\left[\frac{1}{2} - u(j)\right]^2 - \frac{1}{2}\right\}\right) \tag{4}$$

방정식 (3.4.18)을 풀어서 이론적으로 c_1^*와 c_2^*를 구할 수 있다. 그러나, 이 예제에서는 선형회귀모형추정법을 사용해서 구하기로 하자. 선형회귀모형추정법을 사용할 때 유의할 점은 식 (4)에는 상수항이 포함되지 않는다는 것이다. 식 (4)에서 알 수 있듯이, 다음 식이 성립한다.

$$E\left(\hat{\theta}_{CV}\right) = \theta \tag{5}$$

원시몬테카를로법과 다변량제어변량법을 비교하기 위해서, 다음 MATLAB프로그램 CrudeControl102.m을 실행해 보자.

```
% ------------------------------------------------------------------------
% Filename: CrudeControl102.m
% Crude vs Multiple Control Variate Monte Carlo Integration
% Programmed by CBS
% ------------------------------------------------------------------------
clear all, close all
rand('twister',5489);  % Use twister or state for seed.
Ns = 10000             % Number of Samples
u = rand(Ns,1);
u0 = ones(Ns,1);
u1 = (1-u)-1/2;
d1 = mean(u1)
u2 = (1/2-u).^2-1/12;
d2 = mean(u2)
ucircle = 4*sqrt(1-u.^2);
CovZZX = cov([ u1 u2 ucircle ])
invCovZZ = inv(CovZZX(1:2,1:2))
cstar = invCovZZ*CovZZX(1:2,3)       % Using Least Square Method
cstar2 = regress(ucircle,[u1,u2])    % Using Regression Analysis
for ndum = 1:1:40
    Ndum(ndum) = ndum*200;
    dumMC = ucircle(1:Ndum(ndum));
    CrudeMC(ndum) = mean(dumMC);
    CrudeStd(ndum) = std(dumMC)/sqrt(Ndum(ndum));
    dumMC1 = dumMC - cstar(1)*u1(1:Ndum(ndum));
    ControlMC1(ndum) = mean(dumMC1);
    ControlStd1(ndum) = std(dumMC1)/sqrt(Ndum(ndum));
    VarRatio1(ndum) = ControlStd1(ndum)^2/CrudeStd(ndum)^2;
    dumMC2 = dumMC - cstar(1)*u1(1:Ndum(ndum)) ...
             - cstar(2)*u2(1:Ndum(ndum));
    ControlMC2(ndum) = mean(dumMC2);
    ControlStd2(ndum) = std(dumMC2)/sqrt(Ndum(ndum));
    VarRatio2(ndum) = ControlStd2(ndum)^2/CrudeStd(ndum)^2;
end
% Ploting
subplot(2,1,1)
theta = pi
thet = theta*ones(1,40);
```

```
plot(Ndum,thet,'k:',Ndum,CrudeMC,'r-',Ndum,ControlMC1,'g--', ...
     Ndum,ControlMC2,'b:','LineWidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[3.1,3.2])
legend('True','Crude','CV 1','CV 2','location','SW')
xlabel('\bf Number of Random Numbers','Fontsize',12');
ylabel('\bf MC Integral','Fontsize',12');
hold off
% Ratio of Sample Variances
subplot(2,1,2)
plot(Ndum,VarRatio1,'g--',Ndum,VarRatio2,'b:','LineWidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[-0.05 0.20])
legend('CV 1','CV 2','location','SW')
xlabel('\bf Number of Random Numbers','Fontsize',12');
ylabel('\bf Variance Ratio','Fontsize',12');
saveas(gcf,'CrudeControl102','epsc')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 10000개의 일양샘플들을 발생시켜서 식 (4)의 계수들을 추정하면 다음과 같다.

$$C_{zz} = \begin{bmatrix} 0.0829 & 0.0002 \\ 0.0002 & 0.0055 \end{bmatrix}, \quad \boldsymbol{c}_{zy} = \begin{bmatrix} 0.2346 \\ -0.0233 \end{bmatrix} \tag{1}$$

$$\boldsymbol{c}^* = C_{zz}^{-1}\boldsymbol{c}_{zx} = \begin{bmatrix} 12.0629 & -0.5367 \\ -0.5367 & 180.8439 \end{bmatrix} \begin{bmatrix} 0.2346 \\ -0.0233 \end{bmatrix} = \begin{bmatrix} 2.8425 \\ -4.3434 \end{bmatrix} \tag{2}$$

식 (2)는 최소제곱추정법에 의한 것이다. MATLAB의 함수 regress.m을 사용해서 $\boldsymbol{c}^*$를 구하면, 다음과 같다.

$$\boldsymbol{c}^* = \begin{bmatrix} 2.8599 \\ -4.5894 \end{bmatrix} \tag{3}$$

식 (3)은 최우추정법에 의한 것이다. 여기서는 최소제곱추정법에 의한 추정값들을 사용하기로 하자.

일양샘플들의 개수를 200에서 8000까지 증가시켜가면서, 모수 θ를 원시몬테카를로법, 1변량 제어변량법, 그리고 2변량 제어변량법으로 추정하기로 하자. 각 방법에 의한 추정량은

다음과 같다.

$$\hat{\theta}_C = \frac{1}{n}\sum_{j=1}^{n} 4\sqrt{1-u(j)^2} \tag{4}$$

$$\hat{\theta}_{CV1} = \frac{1}{n}\sum_{j=1}^{n}\left(4\sqrt{1-u(j)^2} - c_1^*\left[\frac{1}{2}-u(j)\right]\right) \tag{5}$$

$$\hat{\theta}_{CV2} = \frac{1}{n}\sum_{j=1}^{n} v\left(4\sqrt{1-u(j)^2} - c_1^*\left[\frac{1}{2}-u(j)\right] - c_2^*\left\{\left[\frac{1}{2}-u(j)\right]^2 - \frac{1}{1}\right\}\right) \tag{6}$$

이 추정값들이 그림 3.4.7에 그려져 있다. 그림 3.4.7의 상단 그래프에서 적색 실선은 원시몬테카를로법에 의한 추정값 $\hat{\theta}_C$를, 녹색 긴점선은 1변량 제어변량법에 의한 추정값 $\hat{\theta}_{CV1}$을, 청색 점선은 2변량 제어변량법에 의한 추정값 $\hat{\theta}_{CV2}$를 나타낸다. 이 상단 그래프에서 알 수 있듯이, 적절한 추정값들 c_1^*와 c_2^*를 사용하면 원시몬테카를로법에 의한 추정값, 1변량 제어변량법에 의한 추정값, 그리고 2변량 제어변량법에 의한 추정값 순으로 진짜 정적분에 가깝다. 그림 3.4.7의 하단 그래프에서 녹색 긴점선은 원시몬테카를로법에 의한 추정값 $\hat{\theta}_C$에 대한 1변량 제어변량법에 의한 추정값 $\hat{\theta}_{CV1}$의 오차분산비를 나타내고, 청색 점선은 원시몬테카를로법에 의한 추정값 $\hat{\theta}_C$에 대한 2변량 제어변량법에 의한 추정값 $\hat{\theta}_{CV2}$의 표본오차분산비를 나타낸다. 이 하단 그래프에서 알 수 있듯이, 이 예제에서는 2변량 제어변량법이 1변량 제어변량법에 비해 훨씬 효율적이다. ■

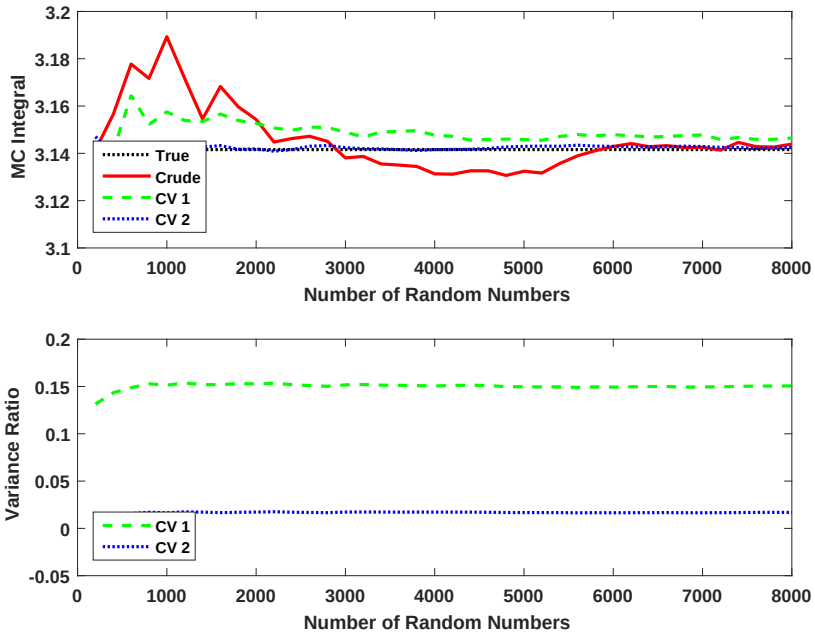


그림 3.4.7. 원시몬테카를로법과 다변량제어변량법 (MATLAB)

예제 3.4.8 R언어를 사용해서 예제 3.4.7의 문제를 풀기 위해서, 다음 R프로그램 CrudeControl102R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: CrudeControl102R.R
# Crude vs Control Variate Monte Carlo Integration 1
# Programmed by CBS
# ------------------------------------------------------------------------
# Modelling
set.seed(11)
Ns <- 10000              # Number of Samples
u <- runif(Ns)
u0 <- rep(1,Ns)
u1 <- (1-u)-1/2;
d1 <- mean(u1)
u2 <- (1/2-u)^2-1/12
d2 <- mean(u2)
ucircle <- 4*sqrt(1-u^2)
( CovZZX <- cov(cbind(u1,u2,ucircle)) )
( invCovZZ <- solve(CovZZX[1:2,1:2]) )
( cstar <- invCovZZ%*%CovZZX[1:2,3] )    # Using Least Square Method
( cstar2 <- glm(ucircle ~ 0+u1+u2) )     # Using Regression w/o constant term

# Generating Random Numbers and Simulations
NoSim <- 40
Ndum <- rep(0,NoSim)
CrudeMC <- CrudeSd <- rep(0,NoSim)
ControlMC1 <- ControlSd1 <- VarRatio1 <- rep(0,NoSim)
ControlMC2 <- ControlSd2 <- VarRatio2 <- rep(0,NoSim)

for( ndum in 1:1:NoSim ){
    Ndum[ndum] <- ndum*200
    dumMC <- ucircle[1:Ndum[ndum]]
    CrudeMC[ndum] <- mean(dumMC)
    CrudeSd[ndum] <- sd(dumMC)/sqrt(Ndum[ndum])
    dumMC1 <- dumMC - cstar[1]*u1[1:Ndum[ndum]]
    ControlMC1[ndum] <- mean(dumMC1)
    ControlSd1[ndum] <- sd(dumMC1)/sqrt(Ndum[ndum])
    VarRatio1[ndum] <- ControlSd1[ndum]^2/CrudeSd[ndum]^2
    dumMC2 <- dumMC - cstar[1]*u1[1:Ndum[ndum]] -
                      cstar[2]*u2[1:Ndum[ndum]]
    ControlMC2[ndum] <- mean(dumMC2)
    ControlSd2[ndum] <- sd(dumMC2)/sqrt(Ndum[ndum])
    VarRatio2[ndum] <- ControlSd2[ndum]^2/CrudeSd[ndum]^2
}

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
thet <- pi*rep(1,NoSim)
postscript('CrudeControl102R.eps')  # Start to save figure
RVdata <- data.frame(thet,CrudeMC,ControlMC1,ControlMC2,VarRatio1,VarRatio2)
plotMC <- ggplot(RVdata,aes(x=Ndum,y=CrudeMC)) +
          geom_line(col="red",lwd=1.2,linetype=1) +
          geom_hline(yintercept=pi,col="dark blue",lwd=0.8) +
          ylim(3.1,3.2)+
```

```
          xlab("Number of Random Numbers") + ylab("MC integral") +
          geom_line(aes(x=Ndum,y=ControlMC1),col="green",lwd=1.2,linetype=2)+
          geom_line(aes(x=Ndum,y=ControlMC2),col="blue",lwd=1.2,linetype=3)
plotVR <- ggplot(RVdata,aes(x=Ndum,y=VarRatio1)) +
          geom_line(col="green",lwd=1.2,linetype=2) +
          geom_hline(yintercept=1,col="dark blue",lwd=0.8) +
          geom_line(aes(x=Ndum,y=VarRatio2),col="blue",lwd=1.2,linetype=3) +
          ylim(-0.05,0.20) +
          xlab("Number of Random Numbers") + ylab("Variance Ratio")
pushViewport(viewport(layout=grid.layout(2,1)))
print(plotMC, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plotVR, vp=viewport(layout.pos.row=2, layout.pos.col=1))
dev.off()            # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 10000개의 일양샘플들을 발생시켜서 식 (4)의 계수들을 추정하면 다음과 같다.

$$C_{zz} = \begin{bmatrix} 0.0825 & -0.0003 \\ -0.0003 & 0.0055 \end{bmatrix}, \quad \boldsymbol{c}_{zy} = \begin{bmatrix} 0.2366 \\ -0.0252 \end{bmatrix} \tag{1}$$

$$\boldsymbol{c}^* = C_{zz}^{-1}\boldsymbol{c}_{zx} = \begin{bmatrix} 12.1236 & 0.6514 \\ 0.6514 & 182.2100 \end{bmatrix} \begin{bmatrix} 0.2366 \\ -0.0252 \end{bmatrix} = \begin{bmatrix} 2.8515 \\ -4.4420 \end{bmatrix} \tag{2}$$

식 (2)는 최소제곱추정법에 의한 것이다. R의 함수 glm을 사용해서 $\boldsymbol{c}^*$를 구하면, 다음과 같다.

$$\boldsymbol{c}^* = \begin{bmatrix} 2.682 \\ -4.922 \end{bmatrix} \tag{3}$$

식 (3)은 최우추정법에 의한 것이다. 여기서는 최소제곱추정법에 의한 추정값들을 사용하기로 하자.

일양샘플들의 개수를 200에서 8000까지 증가시켜가면서, 모수 θ를 원시몬테카를로법, 1변량 제어변량법, 그리고 2변량 제어변량법으로 추정하기로 하자. 이 모수 θ의 추정값들이 그림 3.4.8에 그려져 있다. 그림 3.4.8의 상단 그래프에서 적색 실선은 원시몬테카를로법에 의한 추정값 $\hat{\theta}_C$를, 녹색 긴점선은 1변량 제어변량법에 의한 추정값 $\hat{\theta}_{CV1}$을, 청색 점선은 2변량 제어변량법에 의한 추정값 $\hat{\theta}_{CV2}$를 나타낸다. 이 상단 그래프에서 알 수 있듯이, 적절한 추정값들 c_1^*와 c_2^*를 사용하면 원시몬테카를로법에 의한 추정값, 1변량 제어변량법에 의한 추정값, 그리고 2변량 제어변량법에 의한 추정값 순으로 진짜 정적분에 가깝다. 그림 3.4.8의 하단 그래프에서 녹색 긴점선은 원시몬테카를로법에 의한 추정값 $\hat{\theta}_C$에 대한 1변량 제어변량법에 의한 추정값 $\hat{\theta}_{CV1}$의 오차분산비를 나타내고, 청색 점선은 원시몬테카를로법에 의한

추정값 $\hat{\theta}_C$에 대한 2변량 제어변량법에 의한 추정값 $\hat{\theta}_{CV2}$의 표본오차분산비를 나타낸다. 이 하단 그래프에서 알 수 있듯이, 이 예제에서는 2변량 제어변량법이 1변량 제어변량법에 비해 훨씬 효율적이다. ■

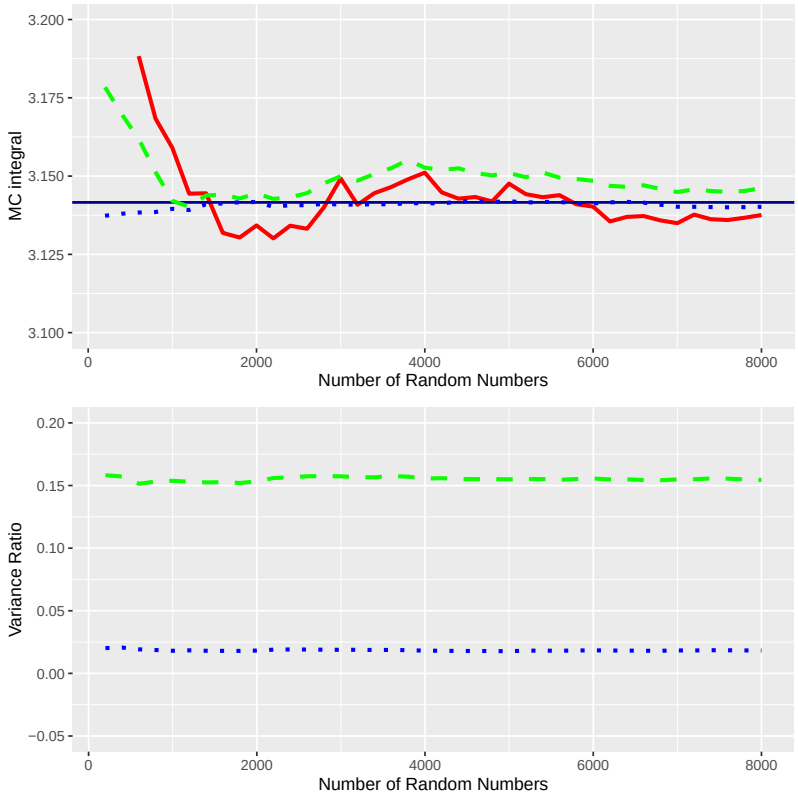


그림 3.4.8. 원시몬테카를로법과 다변량제어변량법 (R)

다음 예제는 대조변량법과 제어변량법을 비교하기 위한 것이다.

예제 3.4.9 다음 정적분을 추정하기로 하자.

$$\theta \doteq \int_0^{\pi/4} \int_0^{\pi/4} x^2 y^2 \sin(x+y) \ln(x+y) dx dy \tag{1}$$

다음과 같은 변수변환을 하자.

$$u_1 = \frac{x}{\pi/4}, \quad u_2 = \frac{y}{\pi/4} \tag{2}$$

다음 함수를 정의하자.

$$f(x,y) \doteq x^2 y^2 \sin(x+y)\ln(x+y) \tag{3}$$

식 (1)을 다음과 같이 쓸 수 있다.

$$\theta = \frac{\pi^2}{16}\int_0^1\int_0^1 f\left(\frac{\pi}{4}u_1, \frac{\pi}{4}u_2\right) du_1 du_2 \tag{4}$$

지지대가 $(0,1)$이고 서로 독립인 일양샘플들 $\{u_{1,j} \mid j=1,2,\cdots,n\}$와 $\{u_{2,j} \mid j=1,2,\cdots,n\}$로 구성된 일양샘플벡터들 $\left\{\boldsymbol{u(j)} = [u_{1,j}, u_{2,j}]^t\right\}$을 발생시키면, 원시몬테카를로법에 의한 추정값은 다음과 같다.

$$\hat{\theta}_C = \frac{\pi^2}{16}\frac{1}{n^2}\sum_{j=1}^{n}\sum_{k=1}^{n} f\left(\frac{\pi u_{1,j}}{4}, \frac{\pi u_{2,k}}{4}\right) \tag{5}$$

대조변량법에 의한 정적분 θ의 추정값은 다음과 같다.

$$\hat{\theta}_A = \frac{\pi^2}{16}\frac{2}{n^2}\sum_{j=1}^{n/2}\sum_{k=1}^{n/2}\left[f\left(\frac{\pi u_{1,j}}{4}, \frac{\pi u_{2,k}}{4}\right) + f\left(\frac{\pi\left[1-u_{1,j}\right]}{4}, \frac{\pi}{4}\left[1-u_{2,k}\right]\right)\right] \tag{6}$$

다음 제어변량함수를 사용하기로 하자.

$$g(x,y) \doteq x^2 y^2 \tag{7}$$

다음 식이 성립함을 쉽게 알 수 있다.

$$\int_0^{\pi/4} x^2 y^2 dxdy = \frac{1}{9}\left[\frac{\pi}{4}\right]^6 \tag{8}$$

제어변량법에 의한 추정값은 다음과 같다.

$$\hat{\theta}_{CV} = \frac{\pi^2}{16}\frac{1}{n^2}\sum_{j=1}^{n}\sum_{k=1}^{n}\left\{f\left(\frac{\pi u_{1,j}}{4}, \frac{\pi u_{2,k}}{4}\right) - \hat{\beta}\left[g\left(\frac{\pi u_{1,j}}{4}, \frac{\pi u_{2,k}}{4}\right) - \frac{1}{9}\frac{\pi^4}{4^4}\right]\right\} \tag{9}$$

여기서 $\hat{\beta}$는 다음과 같다.

$$\hat{\beta} = \frac{\sum_{j=1}^{n}\sum_{k=1}^{n}\left[f\left(\frac{\pi u_{1,j}}{4}, \frac{\pi u_{2,k}}{4}\right) - \bar{f}\right]\left[g\left(\frac{\pi u_{1,j}}{4}, \frac{\pi u_{2,k}}{4}\right) - \bar{g}\right]}{\sum_{j=1}^{n}\sum_{k=1}^{n}\left[g\left(\frac{\pi u_{1,j}}{4}, \frac{\pi u_{2,k}}{4}\right) - \bar{g}\right]^2} \tag{10}$$

단, $\bar{f}$와 $\bar{g}$는 각각 다음과 같다.

$$\bar{f} \doteq \frac{1}{n^2}\sum_{j=1}^{n}\sum_{k=1}^{n} f\left(\frac{\pi u_{1,j}}{4}, \frac{\pi u_{2,k}}{4}\right) \tag{11}$$

$$\bar{g} \doteq \frac{1}{n^2}\sum_{j=1}^{n}\sum_{k=1}^{n} g\left(\frac{\pi u_{1,j}}{4}, \frac{\pi u_{2,k}}{4}\right) \tag{12}$$

원시몬테카를로법, 대조변량법, 그리고 제어변량법을 비교하기 위해서, 다음 MATLAB 프로그램 AntitheticControl101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
% Filename: AntitheticControl101.m
% Antithetic vs Control Variate Integration 1
% Programmed by SBS
% ------------------------------------------------------------------------
clear all, close all, format long
syms x y
theta = int(int(x.^2.*y.^2.*sin(x+y).*log(x+y),x,0,pi/4),y,0,pi/4)
theta = double(theta)
% Monte Sarlo integration
rand('twister',5489);   % Use twister or state for seed.
N = 40000;
u = rand(2,N);
ftn = @(x) prod(x).^2.*sin(sum(x)).*log(sum(x))
% Crude MC
Cr = pi^2*ftn(pi*u/4)/16;
disp([mean(Cr) std(Cr) 2*std(Cr)/sqrt(N)])
% Antithetic MC
u05 = u(:,1:N/2);
Ca = pi^2*( (ftn(pi*u05/4)+ftn(pi*(1-u05)/4))/2 )/16;
disp([mean(Ca) std(Ca) ])
% Control-Variate MC
ff = pi^2*prod(pi*u/4).^2/16;
ffbar = mean(ff)
gg = pi^2*ftn(pi*u/4)/16;
ggbar = mean(gg)
beta =  sum((gg-ggbar).*(ff-ffbar))/sum((ff-ffbar).^2)
ffmu = (pi/4)^6/9
Cc = gg - beta*(ff-ffmu);
disp([mean(Cc) std(Cc) ])
save('Antithetic_Control101','Cr','Ca','Cc')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 수치적분을 사용해서 다음 정적분 θ의 값을 출력한다.

$$\theta = 0.0039180678 \tag{13}$$

일양샘플벡터들을 40000개 발생시켜서 구한 원시몬테카를로법에 의한 추정값은 다음과 같다.

$$\hat{\theta}_C = 0.0038967892 \tag{14}$$

이 추정량의 표본표준편차는 0.0127827785이다.

대조변량법을 사용해서 구한 추정값은 다음과 같다.

$$\hat{\theta}_A = 0.0039285684 \tag{15}$$

이 추정량의 표본표준편차는 0.0085859590이다.

제어변량법을 적용하기 위해서, 식 (10)의 계수 $\hat{\beta}$를 구하기로 하자. 이 MATLAB프로그램을 실행하면, 다음 값들을 얻는다.

$$\bar{f} = 0.0259896644 \;\; \hat{g} = 0.0038967892, \;\; E(f) = 0.026079351 \tag{16}$$

따라서, 계수 $\hat{\beta}$는 다음과 같다.

$$\hat{\beta} = 0.3029455836 \tag{17}$$

이 값들과 제어변량식 (9)를 사용해서 구한 추정값은 다음과 같다.

$$\hat{\theta}_C = 0.0039239593 \tag{18}$$

이 추정량의 표본표준편차는 0.0049083678이다.

결론적으로, 이 예제에서는 제어변량법에 의한 추정값이 가장 정적분에 가깝고 다음이 대조변량법에 의한 추정값이다. 또한, 오차표준편차 역시 제어변량법에 의한 값이 가장 작고 다음이 대조변량법에 의한 값이다. 따라서, 이 예제에서는 제어변량법이 가장 우수하고 그 다음이 대조변량법이다. ■

예제 3.4.10 R언어를 사용해서 예제 3.4.9의 문제를 풀기 위해서, 다음 R프로그램 AntitheticControl101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: AntitheticControl101R.R
# Antithetic vs Control Variate Monte Carlo Integration
# Programmed by CBS
# ------------------------------------------------------------------------
# True Integration
ftn <- function(x,y) {
           Integrand <- x^2*y^2*sin(x+y)*log(x+y)
           return(Integrand)
           }
thetaa <- integrate(function(y) {
            sapply(y, function(y) {
              integrate(function(x) ftn(x,y), 0, pi/4)$value
            })
          }, 0, pi/4)
thetaa


# Monte Sarlo integration
set.seed(11)
N <- 40000;
u <- runif(2*N)
dim(u) <- c(2,N)

## Crude MC
Cr <- rep(0,N)
for (ii in 1:N){
  Cr[ii] <- pi^2/16*ftn(pi/4*u[1,ii],pi/4*u[2,ii])
}
( mean(Cr) ); ( sd(Cr) ); (2*sd(Cr)/sqrt(N) )

## Antithetic MC
u05 <- u[,1:N/2]
Ca <- pi^2*( (ftn(pi*u05[1,]/4,pi*u05[2,]/4) +
               ftn(pi*(1-u05[1,])/4,pi*(1-u05[2,])/4) )/2 )/16;
( mean(Ca) ); ( sd(Ca) ); ( 2*sd(Ca)/sqrt(N) )

##  Control-Variate MC
ff <- pi^2/16*(pi/4*u[1,]*pi/4*u[2,])^2
( ffbar <- mean(ff) )
gg <- pi^2*ftn(pi*u[1,]/4,pi*u[2,]/4)/16;
( ggbar <- mean(gg) )
( beta <-  sum((gg-ggbar)*(ff-ffbar))/sum((ff-ffbar)^2) )
 ( ffmu <- (pi/4)^6/9 )
Cc <- gg - beta*(ff-ffmu)
( mean(Cc) ); ( sd(Cc) ); ( 2*sd(Cc)/sqrt(N) )
# ------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 수치적분을 사용해서 다음 정적분 θ의 값을 출력한다.

$$\theta = 0.003918068 \tag{1}$$

여기서 오차의 절대값은 2×10^{-16} 보다 작다.

일양샘플벡터들을 40000개 발생시켜서 구한 원시몬테카를로법에 의한 추정값은 다음과 같다.

$$\hat{\theta}_C = 0.004040979 \tag{2}$$

이 추정량의 표본표준편차는 0.01307259이다.

대조변량법을 사용해서 구한 추정값은 다음과 같다.

$$\hat{\theta}_A = 0.003924003 \tag{3}$$

이 추정량의 표본표준편차는 0.008612045이다.

제어변량법을 적용하기 위해서, 계수 $\hat{\beta}$를 구하기로 하자. 이 R프로그램을 실행하면, 다음 값들을 얻는다.

$$\bar{f} = 0.02639909 \;\; \hat{g} = 0.004040979, \;\; E(f) = 0.02607935 \tag{4}$$

따라서, 계수 $\hat{\beta}$는 다음과 같다.

$$\hat{\beta} = 0.3029455836 \tag{5}$$

이 값들과 제어변량식을 사용해서 구한 추정값은 다음과 같다.

$$\hat{\theta}_C = 0.003943146 \tag{6}$$

이 추정량의 표본표준편차는 $4.968223 \cdot 10^{-5}$ 이다.

결론적으로, 이 예제에서는 제어변량법에 의한 추정값이 가장 정적분에 가깝고 다음이 대조변량법에 의한 추정값이다. 또한, 오차표준편차 역시 제어변량법에 의한 값이 가장 작고 다음이 대조변량법에 의한 값이다. 따라서, 이 예제에서는 제어변량법이 가장 우수하고 그 다음이 대조변량법이다. ■

3.4.3 조건부 몬테카를로법

확률변수들 x와 y의 결합확률밀도함수를 $f(x,y)$라 하고, x와 y의 주변확률밀도함수들을 각각 $f_x(x)$와 $f_y(y)$로 표기하자. 또한, y가 주어졌을 때 x의 조건부확률밀도함수를 $f_x(x \mid y)$로 표기하자.

명제 3.4.1

a) $E_y(E_x(x \mid y)) = E_x(x)$

b) $Var_x(x) = E_y(Var_x(x \mid y)) + Var_y(E_x(x \mid y))$

증명. 조건부기대값에 대해서 다음 식들이 성립함을 알 수 있다.

$$\begin{aligned} E_y(E_x(x \mid y)) &= \int_{-\infty}^{\infty} E_x(x \mid y) f_y(y) dy \\ &= \int_{-\infty}^{\infty} \left(\int_{-\infty}^{\infty} x f_x(x \mid y) dx \right) f_y(y) dy = \int_{-\infty}^{\infty} \left\{ \int_{-\infty}^{\infty} \frac{x f(x,y)}{f_y(y)} dx \right\} f_y(y) dy \\ &= \int_{-\infty}^{\infty} x \left(\int_{-\infty}^{\infty} f(x,y) dy \right) dx = \int_{-\infty}^{\infty} x f_x(x) dx = E_x(x) \end{aligned} \tag{1}$$

조건부분산에 대해서 다음 식이 성립한다.

$$Var_x(x \mid y) = E_x(x^2 \mid y) - [E_x(x \mid y)]^2 \tag{2}$$

식 (2)의 양변을 y에 관한 기대값을 취하면, 다음 식이 성립함을 알 수 있다.

$$E_x(Var_x(x \mid y)) = E_x(x^2) - E_y\left([E_x(x \mid y)]^2\right) \tag{3}$$

또한, 다음 식이 성립한다.

$$Var_x(E_x(x \mid y)) = E_y\left([E_x(x \mid y)]^2\right) - [E_x(x)]^2 \tag{4}$$

식 (3)과 식 (4)에서 알 수 있듯이, 다음 식들이 성립한다.

$$Var_x(x) = E_x\left(x^2\right) - [E_x(x)]^2 = E_y(Var_x(x \mid y)) + Var_y(E_x(x \mid y)) \tag{5}$$

■

명제 3.4.1에서 알 수 있듯이, 다음 식이 성립한다.

$$Var_x(x) \geq Var_y(E_x(x \mid y)) \tag{3.4.22}$$

즉, y가 주어졌을 때 확률변수 x의 조건부기대값을 이용한 분산이 확률변수 x자체의 분산보다 작다. 따라서, 주변확률밀도함수 $f_y(y)$에 따르는 샘플들 $y_1, y_2, \cdots, y_n$을 이용해서, 확률변수 x의 기대값 $\theta = E(x)$를 다음과 같이 추정한다.

$$\hat{\theta} = \frac{1}{n} \sum_{j=1}^{n} E_x \left(x \mid y(j) \right) \tag{3.4.23}$$

이러한 분산감소법을 조건부몬테카를로법(conditional Monte Carlo method) 또는 회귀분석법(reression method)이라 한다. 조건부몬테카를로법은 제어변량법을 확장한 것이다.

예제 3.4.11 서로 독립인 확률변수들 z와 y가 다음 확률분포들을 따른다고 하자.

$$z \overset{d}{\sim} Exponential(1), \quad y \overset{d}{\sim} Exponential\left(\frac{1}{2}\right) \tag{1}$$

여기서 $Exponential(\mu)$는 평균이 μ이고 비율모수(rate)가 $\lambda = 1/\mu$인 지수확률분포를 의미함을 상기하라. 주어진 양수 k에 대해서 확률 $\theta \doteq Pr(z + y > k)$를 추정하기로 하자.

첫째, 원시몬테카를로법으로 θ를 추정하기 위해서는 먼저 확률분포 $Exponential(1)$로부터 지수샘플들 $z_1, z_2, \cdots, z_n$을 생성하고 이와는 독립적으로 확률분포 $Exponential(1/2)$로부터 지수샘플들 $y_1, y_2, \cdots, y_n$를 생성한다. 이들을 사용해서 다음과 같이 원시몬테카를로법에 의한 추정값을 구한다.

$$\hat{\theta}_C = \frac{1}{n} \sum_{j=1}^{n} 1\left(z(j) + y(j) > k\right) \tag{2}$$

둘째, 조건부몬테카를로법으로 θ를 추정하기로 하고, 다음 확률변수를 정의하자.

$$x \doteq 1\left(z + y > k\right) \tag{3}$$

다음 식들이 성립한다.

$$E_x\left(x \mid y\right) = P\left(z + y > k \mid y\right) = P\left(z > k - y\right) = 1 - F_z(k - y) \tag{4}$$

여기서 $F_z(\cdot)$는 확률변수 z의 확률분포함수이다. 다음 식이 성립한다.

$$1 - F_z(k - y) = \begin{cases} e^{-[k-y]}, & (0 \le y \le k) \\ 1, & (y > k) \end{cases} \tag{5}$$

식 (4)와 식 (5)에서 알 수 있듯이, 다음 식이 성립한다.

$$E_x\left(x \mid y\right) = \begin{cases} e^{-[k-y]}, & (0 \le y \le k) \\ 1, & (y > k) \end{cases} \tag{6}$$

따라서, 조건부몬테카를로법에 의한 추정값은 다음과 같다.

$$\hat{\theta}_{CM} = \frac{1}{n}\sum_{j=1}^{n} E_x\left(x \mid y(j)\right) \tag{7}$$

식 (6)과 반복조건부기대값법칙을 사용해서, 확률 θ의 참값을 다음과 같이 구할 수 있다. 다음 식들이 성립한다.

$$\theta = E\left(1\left(z + y > k\right)\right) = E_x(x) = E_y\left(E_x\left(x \mid y\right)\right) \tag{8}$$

식 (6)과 식 (8)에서 알 수 있듯이, 다음 식들이 성립한다.

$$\begin{aligned} \theta &= \int_0^k e^{-[k-y]} \cdot 2e^{-2y} dy + \int_k^\infty 1 \cdot 2e^{-2y} dy \\ &= 2e^{-k}\left[-e^{-y}\right]_0^k + \left[-e^{-2y}\right]_k^\infty = 2e^{-k} - e^{-2k} \end{aligned} \tag{9}$$

원시몬테카를로법과 조건부몬테카를로법을 비교하기 위해서, 다음 MATLAB프로그램 CrudeConditional101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
% Filename: Crude_ConditionalMC101.m
% Crude vs Conditional Monte Carlo Integrations
% Programmed by CBS
```

```
% ------------------------------------------------------------------------
clear all, close all
rand('twister',5489);   % Use twister or state for seed.
K = 3
Ns = 10000
z = random('Exponential',1,Ns,1);
y = random('exp',1/2,Ns,1);
x = (z+y > K);
for ndum = 1:1:40
    Ndum(ndum) = ndum*200;
    dumMC = x(1:Ndum(ndum));
    CrudeMC(ndum) = mean(dumMC);
    CrudeStd(ndum) = std(dumMC)/sqrt(Ndum(ndum));
    dumy = y(1:Ndum(ndum));
    dumCMC = ones(Ndum(ndum),1)+(exp(dumy-K)-1) ...
              .*(dumy <= K).*(dumy >= 0);
    ConditionMC(ndum) = mean(dumCMC);
    ConditionStd(ndum) = std(dumCMC)/sqrt(Ndum(ndum));
    VarRatio(ndum) = ConditionStd(ndum)^2/CrudeStd(ndum)^2;
end
% Ploting
subplot(2,1,1)
TrueInt = 2*exp(-K)-exp(-2*K)
TI = TrueInt*ones(1,40);
plot(Ndum,TI,'k:',Ndum,CrudeMC,'r-',Ndum,ConditionMC,'g--', ...
     'LineWidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[0.08 0.1])
legend('True Value','Crude MC','Conditional MC','location','SE')
xlabel('\bf Number of Random Numbers','Fontsize',12');
ylabel('\bf MC Integral','Fontsize',12');
hold off
subplot(2,1,2)
plot(Ndum,VarRatio,'g--','LineWidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[0 0.12])
xlabel('\bf Number of Random Numbers','Fontsize',12');
ylabel('\bf Variance Ratio','Fontsize',12');
saveas(gcf,'CrudeConditional101','epsc')
save('CrudeConditionalMC101','CrudeMC','ConditionMC')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 확률 $\theta = Pr\,(z+y>3)$를 원시몬테카를로법과 조건부몬테카를로법으로 추정한다.

지수샘플벡터들의 개수를 200에서 8000까지 증가시켜가면서, 모수 θ를 원시몬테카를로법과 조건부몬테카를로법으로 추정하기로 하자. 이 추정값들이 그림 3.4.9 에 그려져 있다. 그림 3.4.9 의 상단 그래프에서 적색 실선은 원시몬테카를로법에 의한 추정값 $\hat{\theta}_C$를, 그리고 녹색 긴점선은 조건부몬테카를로법에 의한 추정값 $\hat{\theta}_{CM}$을 나타낸다. 이 그래프에서 알 수 있듯이, 원시몬테카를로법에 의한 추정값보다 조건부몬테카를로법에 의한 추정값이 진짜 확률에 빨리 수렴한다. 그림 3.4.9 의 하단 그래프에서 적색 긴점선은 원시몬테카를로법에 의한 추정값 $\hat{\theta}_C$에 대한 조건부몬테카를로법에 의한 추정값 $\hat{\theta}_{CM}$의 오차분산비를 나타낸다. 이 그래프에서 알 수 있듯이, 이 예제에서는 2변량 조건부몬테카를로법이 원시몬테카르로법에 비해 훨씬

효율적이다. ■

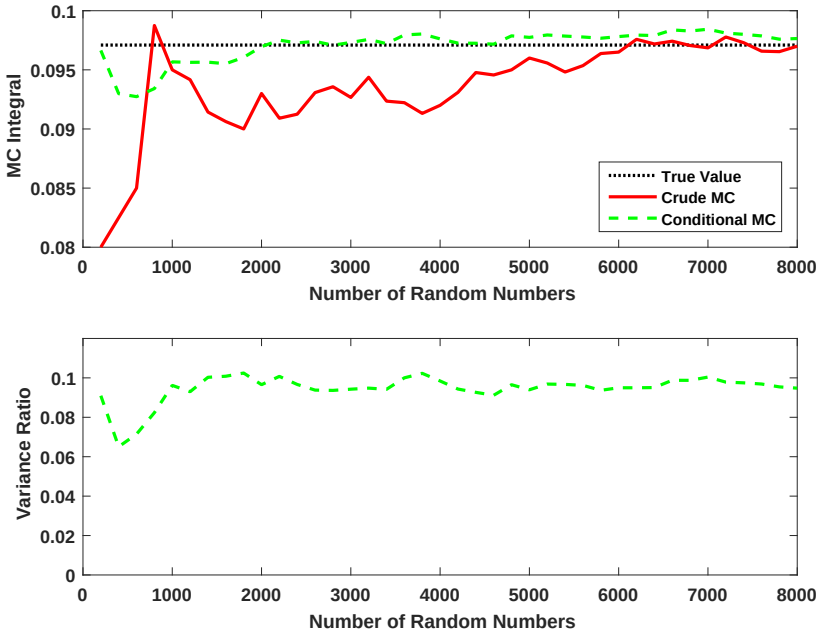


그림 3.4.9. 원시몬테카를로법과 조건부몬테카를로법 1 (MATLAB)

예제 3.4.12 R언어를 사용해서 예제 3.4.9의 문제를 풀기 위해서, 다음 R프로그램 Crude-Conditional101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------------
# Filename: CrudeConditional101R.R
# Crude vs Conditional Monte Carlo
# Programmed by CBS
# ------------------------------------------------------------------------------
# Simulation
set.seed(11)
K <- 3
Ns <- 10000
z <- rexp(Ns,1)
y <- rexp(Ns,2)
x <- 0+(z+y > K)
NoSim <- 40
Ndum <- rep(0,NoSim)
CrudeMC <- CrudeSd  <- rep(0,NoSim)
ConditionalMC <- ConditionalSd  <- rep(0,NoSim)
VarRatio <- rep(0,NoSim)

for( ndum in 1:1:NoSim ){
    Ndum[ndum] <- ndum*200
    dumMC <- x[1:Ndum[ndum]]
    CrudeMC[ndum] <- mean(dumMC)
    CrudeSd[ndum] <- sd(dumMC)/sqrt(Ndum[ndum])
    dumy <- y[1:Ndum[ndum]]
    dumCMC <- rep(1,Ndum[ndum])+(exp(dumy-K)-1)*(dumy <= K)*(dumy >= 0)
```

```
    ConditionalMC[ndum] <- mean(dumCMC)
    ConditionalSd[ndum] <- sd(dumCMC)/sqrt(Ndum[ndum])
    VarRatio[ndum] <- ConditionalSd[ndum]^2/CrudeSd[ndum]^2
}
VarRatio

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('CrudeConditional101R.eps')  # Start to save figure
thetaa <- 2*exp(-K)-exp(-2*K)
RVdata <- data.frame(CrudeMC,ConditionalMC,VarRatio)
plotMC <- ggplot(RVdata,aes(x=Ndum,y=CrudeMC)) +
          geom_point(shape=16,col="red",lwd=1.7) +
          geom_line(col="red",linetype=1) +
          geom_point(aes(x=Ndum,y=ConditionalMC),shape=16,col="green",lwd=1.7)+
          geom_line(aes(x=Ndum,y=ConditionalMC),col="green",linetype=2) +
          geom_hline(yintercept=thetaa,col="dark blue",lwd=1) +
          ylim(0.08,0.1) +
          xlab("Number of Samples") + ylab("MC estimate")
plotVR <- ggplot(RVdata,aes(x=Ndum,y=VarRatio)) +
          geom_point(shape=16,col="green",lwd=1.7) +
          geom_line(col="green",linetype=2) +
          # ylim(0,0.12) +
          xlab("Number of Samples") + ylab("Variance Ratio")
pushViewport(viewport(layout=grid.layout(2,1)))
print(plotMC, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plotVR, vp=viewport(layout.pos.row=2, layout.pos.col=1))
dev.off()            # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 확률 $\theta = Pr(z+y>3)$를 원시몬테카를로법과 조건부몬테카를로법으로 추정한다.

지수샘플벡터들의 개수를 200에서 8000까지 증가시켜가면서, 모수 θ를 원시몬테카를로법과 조건부몬테카를로법으로 추정하기로 하자. 이 추정값들이 그림 3.4.10 에 그려져 있다. 그림 3.4.10 의 상단 그래프에서 적색 실선은 원시몬테카를로법에 의한 추정값 $\hat{\theta}_C$를, 그리고 녹색 긴점선은 조건부몬테카를로법에 의한 추정값 $\hat{\theta}_{CM}$을 나타낸다. 이 그래프에서 알 수 있듯이, 원시몬테카를로법에 의한 추정값보다 조건부몬테카를로법에 의한 추정값이 진짜 확률에 빨리 수렴한다. 그림 3.4.10 의 하단 그래프에서 적색 긴점선은 원시몬테카를로법에 의한 추정값 $\hat{\theta}_C$에 대한 조건부몬테카를로법에 의한 추정값 $\hat{\theta}_{CM}$의 오차분산비를 나타낸다. 이 그래프에서 알 수 있듯이, 이 예제에서는 2변량 조건부몬테카를로법이 원시몬테카르로법에 비해 훨씬 효율적이다. ■

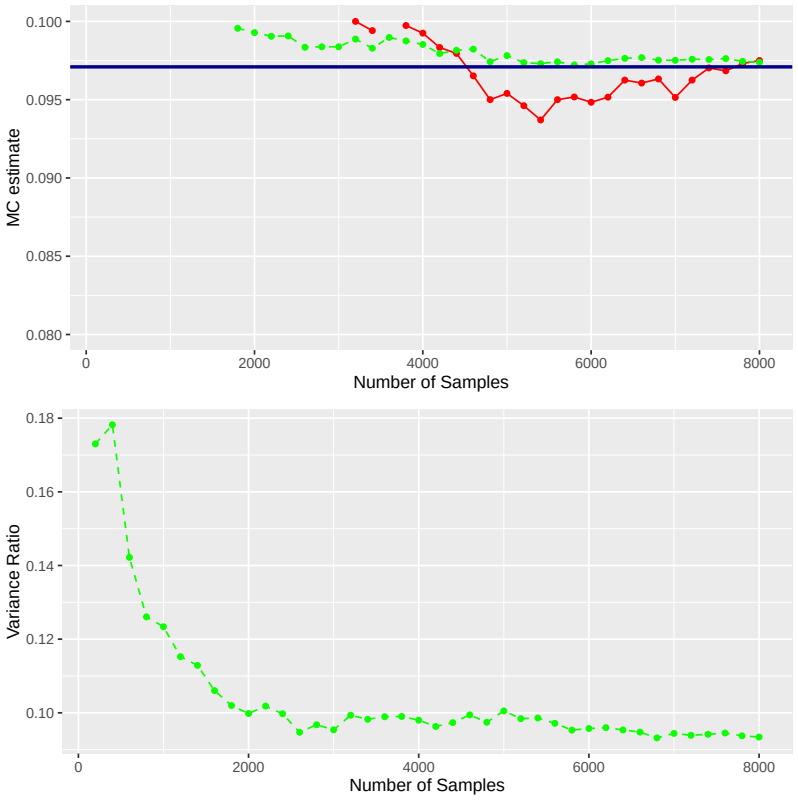


그림 3.4.10. 원시몬테카를로법과 조건부몬테카를로법 1 (R)

예제 3.4.13 모수들이 r과 p인 음이항확률변수 x의 확률밀도함수 $f(x)$는 다음과 같다.

$$f_x(x) = \binom{r+x-1}{r-1} p^r [1-p]^x, \quad (x = 0, 1, \cdots) \tag{1}$$

이 예제에서는 모수들이 $r = 3$과 $p = 0.5$인 음이항확률변수 x에 대해서 다음과 같은 기대값 θ를 계산해 보자.

$$\theta = E(x) = \sum_{x=0}^{\infty} x \binom{r+x-1}{r-1} p^r [1-p]^x \tag{2}$$

이 음이항확률분포의 평균과 분산은 각각 다음과 같다.

$$E(x) = \frac{r[1-p]}{p}, \quad Var(x) = \frac{r[1-p]}{p^2} \tag{3}$$

따라서, 다음 식들이 성립한다.

$$E(x) = 3, \quad Var(x) = 6 \tag{4}$$

첫째, 원시몬테카를로법으로 θ를 추정하기 위해서는 먼저 모수들이 $r = 3$과 $p = 0.5$인 음이항확률분포에서 샘플들 $x_1, x_2, \cdots, x_n$을 생성한다. 이들을 사용해서 다음과 같이 원시몬테카를로법에 의한 추정값을 구한다.

$$\hat{\theta}_C = \frac{1}{n}\sum_{j=1}^{n} x(j) \tag{5}$$

식 (4)에서 알 수 있듯이, 다음 식이 성립한다.

$$Var\left(\hat{\theta}_C\right) = \frac{6}{n} \tag{6}$$

둘째, 조건부몬테카를로법으로 θ를 추정하기로 하자. 확률변수 y는 다음 감마확률분포를 따른다고 하자.

$$y \overset{d}{\sim} Gamma(r, b) \tag{7}$$

이 감마분포의 형태모수(shape parameter)는 r이고, 척도모수(scale parameter)는 b이고, 비율모수는 $1/b$이다. 따라서, 평균은 rb이고 분산은 rb^2이다. 조건부확률변수 $z \mid y$가 모수가 y인 Poisson확률분포를 따르면, 이에 해당하는 조건부확률밀도함수는 다음과 같다.

$$f_{z\mid y}(z) = e^{-y}\frac{y^z}{z!}, \ \ (z = 0, 1, \cdots) \tag{8}$$

다음 식들이 성립한다.

$$\begin{aligned} f_z(z) &= \int_0^\infty f_{z\mid y}(z) f(y) dy \\ &= \int_0^\infty e^{-y}\frac{y^z}{z!}\frac{1}{\Gamma(r) b^r y^{r-1}} \exp\left(-\frac{y}{b}\right) dy \\ &= \frac{1}{\Gamma(r) b^r z!}\int_0^\infty y^{z+r-1} \exp\left(-\left[1+\frac{1}{b}\right] y\right) dy \\ &= \binom{z+r-1}{z}\frac{1}{[b+1]^r}\left[\frac{b}{1+b}\right]^z \end{aligned} \tag{9}$$

식 (9)에서 알 수 있듯이, z는 모수들이 r과 $p = 1/[1+b]$인 음이항확률변수이다. 따라서, 만약 $b = 1$이면, z는 모수들이 $r = 3$과 $p = 0.5$인 음이항확률변수이다. 식 (8)에서 알 수 있듯이,

조건부확률변수 $z \mid y$가 모수가 y인 Poisson확률분포를 따르면, 다음 식들이 성립한다.

$$E_z(z \mid y) = \sum_{z=0}^{\infty} f_{z \mid y}(z) = \sum_{z=0}^{\infty} z e^{-y} \frac{y^z}{z!} = y \tag{10}$$

식 (10)에서 알 수 있듯이, 모수들이 $r = 3$과 $b = 1$인 감마확률분포에서 생성된 감마샘플들을 $y_1, y_2, \cdots, y_n$라고 하면, 조건부몬테카를로법에 의한 θ의 추정값은 다음과 같다.

$$\hat{\theta}_{CM} = \frac{1}{n} \sum_{j=1}^{n} E_z\left(z \mid y(j)\right) = \frac{1}{n} \sum_{j=1}^{n} y(j) \tag{11}$$

식 (11)에서 알 수 있듯이, 다음 식이 성립한다.

$$Var\left(\hat{\theta}_{CM}\right) = \frac{3 \cdot 1^2}{n} = \frac{3}{n} \tag{12}$$

식 (12)의 조건부몬테카를로법에 의한 분산과 식 (6)의 원시몬테카를로법에 의한 분산의 비, 즉 오차분산비는 0.5이다.

원시몬테카를로법과 조건부몬테카를로법을 비교하기 위해서, 다음 MATLAB프로그램 CrudeConditional102.m을 실행해 보자.

```
% ------------------------------------------------------------------------
% Filename: CrudeConditional102.m
% Crude vs Conditional Monte Carlo Integrations 2
% Programmed by CBS
% ------------------------------------------------------------------------
clear all, close all
rand('twister',5489);   % Use twister or state for seed.
r = 3, p = 1/2, b = 1/p -1
theta = r
Ns = 10000
x = random('nbin',r,p,1,Ns);
y = random('gam',r,b,1,Ns);
stepS = Ns/40
for ndum = 1:1:40
    Ndum(ndum) = stepS*ndum;
    dumMC = x(1:Ndum(ndum));
    CrudeMC(ndum) = mean(dumMC);
    CrudeStd(ndum) = std(dumMC)/sqrt(Ndum(ndum));
    dumCMC = y(1:Ndum(ndum));
    ConditionMC(ndum) = mean(dumCMC);
    ConditionStd(ndum) = std(dumCMC)/sqrt(Ndum(ndum));
    VarRatio(ndum) = ConditionStd(ndum)^2/CrudeStd(ndum)^2;
end
% Ploting
subplot(2,1,1)
TI = theta*ones(1,40);
plot(Ndum,TI,'k:',Ndum,CrudeMC,'r-', ...
     Ndum,ConditionMC,'g--','LineWidth',2)
```

```
set(gca,'fontsize',11,'fontweigh','bold','ylim',[2.8 3.1])
legend('True Value','Crude MC','Conditional MC','location','SE')
xlabel('\bf Number of Random Numbers','Fontsize',12');
ylabel('\bf MC Integral','Fontsize',12');
hold off
subplot(2,1,2)
TrueVR = 1/2*ones(1,40);
plot(Ndum,TrueVR,'k:',Ndum,VarRatio,'g--','LineWidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[0.45 0.6])
xlabel('\bf Number of Random Numbers','Fontsize',12');
ylabel('\bf Variance Ratio','Fontsize',12');
saveas(gcf,'CrudeConditional102','epsc')
save('CrudeConditional102','CrudeMC','ConditionMC')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 모수들이 $r = 3$과 $p = 0.5$인 음이항확률변수 x의 기대값 $\theta = E(x)$를 원시몬테카를로법과 조건부몬테카를로법으로 추정한다. 즉, 샘플들의 개수를 250에서 10000까지 증가시켜가면서, 모수 θ를 원시몬테카를로법과 조건부몬테카를로법으로 추정한다. 이 추정값들이 그림 3.4.11에 그려져 있다. 그림 3.4.11의 상단 그래프에서 적색 실선은 원시몬테카를로법에 의한 추정값 $\hat{\theta}_C$를, 그리고 녹색 긴점선은 조건부몬테카를로법에 의한 추정값 $\hat{\theta}_{CM}$을 나타낸다. 그림 3.4.11의 하단 그래프에서 녹색 긴점선은 원시몬테카를로법에 의한 추정값 $\hat{\theta}_C$에 대한 조건부몬테카를로법에 의한 추정값 $\hat{\theta}_{CM}$의 오차분산비를 나타낸다. 이 그래프에서 알 수 있듯이, 오차분산비는 0.5에 가깝다. 따라서, 조건부몬테카를로법이 원시몬테카르로법에 비해 효율적이다. ■

예제 3.4.14 R언어를 사용해서 예제 3.4.13의 문제를 풀기 위해서, 다음 R프로그램 CrudeConditional102R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: CrudeConditional102R.R
# Crude vs Conditional Monte Carlo 2
# Programmed by CBS
# ------------------------------------------------------------------------
# Simulation
set.seed(11)
r <- 3; p <- 1/2;
b <- 1/p -1
thetaa <- r
Ns <- 10000
x <- rnbinom(Ns,r,p)
y <- rgamma(Ns,r,b)
stepS <- Ns/40
NoSim <- 40
Ndum <- rep(0,NoSim)
CrudeMC <- CrudeSd  <- rep(0,NoSim)
ConditionalMC <- ConditionalSd  <- rep(0,NoSim)
```

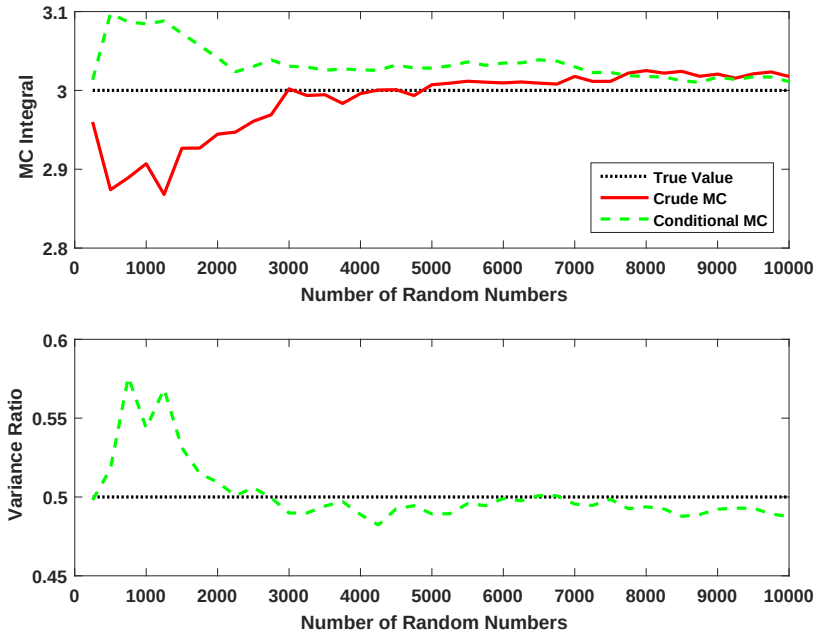


그림 3.4.11. 원시몬테카를로법과 조건부몬테카를로법 2 (MATLAB)

```
VarRatio <- rep(0,NoSim)
for( ndum in 1:1:NoSim ){
    Ndum[ndum] <- ndum*stepS
    dumMC <- x[1:Ndum[ndum]]
    CrudeMC[ndum] <- mean(dumMC)
    CrudeSd[ndum] <- sd(dumMC)/sqrt(Ndum[ndum])
    dumCMC <- y[1:Ndum[ndum]]
    ConditionalMC[ndum] <- mean(dumCMC)
    ConditionalSd[ndum] <- sd(dumCMC)/sqrt(Ndum[ndum])
    VarRatio[ndum] <- ConditionalSd[ndum]^2/CrudeSd[ndum]^2
}
VarRatio

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('CrudeConditional102R.eps')  # Start to save figure
RVdata <- data.frame(CrudeMC,ConditionalMC,VarRatio)
plotMC <- ggplot(RVdata,aes(x=Ndum,y=CrudeMC)) +
          geom_point(shape=16,col="red",lwd=1.7) +
          geom_line(col="red",linetype=1) +
          geom_point(aes(x=Ndum,y=ConditionalMC),shape=16,col="green",lwd=1.7)+
          geom_line(aes(x=Ndum,y=ConditionalMC),col="green",linetype=2) +
          geom_hline(yintercept=thetaa,col="dark blue",lwd=1) +
          ylim(2.8,3.1) +
          xlab("Number of Samples") + ylab("MC estimate")
plotVR <- ggplot(RVdata,aes(x=Ndum,y=VarRatio)) +
          geom_point(shape=16,col="green",lwd=1.7) +
          geom_line(col="green",linetype=2) +
          geom_hline(yintercept=0.5,col="dark blue",lwd=1) +
          ylim(0.45,0.6) +
          xlab("Number of Samples") + ylab("Variance Ratio")
```

```
pushViewport(viewport(layout=grid.layout(2,1)))
print(plotMC, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plotVR, vp=viewport(layout.pos.row=2, layout.pos.col=1))
dev.off()             # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 모수들이 $r = 3$과 $p = 0.5$인 음이항확률변수 x의 기대값 $\theta = E(x)$를 원시몬테카를로법과 조건부몬테카를로법으로 추정한다. 즉, 샘플들의 개수를 250에서 10000까지 증가시켜가면서, 모수 θ를 원시몬테카를로법과 조건부몬테카를로법으로 추정한다. 이 추정값들이 그림 3.4.12에 그려져 있다. 그림 3.4.12의 상단 그래프에서 적색 실선은 원시몬테카를로법에 의한 추정값 $\hat{\theta}_C$를, 그리고 녹색 긴점선은 조건부몬테카를로법에 의한 추정값 $\hat{\theta}_{CM}$을 나타낸다. 그림 3.4.12의 하단 그래프에서 녹색 긴점선은 원시몬테카를로법에 의한 추정값 $\hat{\theta}_C$에 대한 조건부몬테카를로법에 의한 추정값 $\hat{\theta}_{CM}$의 오차분산비를 나타낸다. 이 그래프에서 알 수 있듯이, 오차분산비는 0.5에 가깝다. 따라서, 이 예제에서는 조건부몬테카를로법이 원시몬테카르로법에 비해 효율적이다. ■

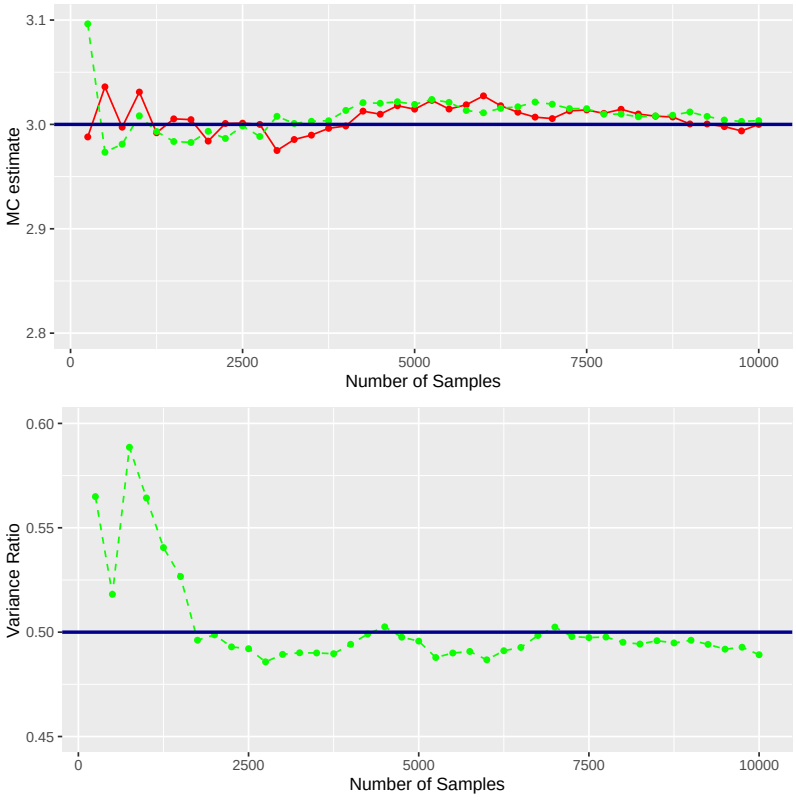


그림 3.4.12. 원시몬테카를로법과 조건부몬테카를로법 2 (R)

3.4.4 층화추출법

층화추출법(stratified sampling method)은 표본론에서 사용되는 대표적인 기법으로서 전체 모집합을 몇 개의 소그룹(subgroup)들로 분할하고, 각 소그룹 별로 표본을 추출하는 방법이다. 예를 들어, 군필자들에 취업가산점을 주는데 대한 남자와 여자의 선호도가 다를 것이다. 따라서, 남녀를 합한 집합에서 표본을 추출해서 찬성율을 조사하는 것보다 남자들로 구성된 소그룹과 여자들로 구성된 소그룹 각각에서 표본을 추출해서 찬성율을 구하는 것이 오차분산의 관점에서 더 효율적일 것이다. 이러한 성질을 이용해서 정적분을 추정하기로 하자. 다음 정적분을 구하기로 하자.

$$\theta = \int_{-\infty}^{\infty} g(x)f(x)dx \tag{3.4.24}$$

이 정적분을 다음과 같이 쓸 수 있다.

$$\theta = \sum_{i=1}^{k} \int_{a_{i-1}}^{a_i} g(x)f(x)dx \tag{3.4.25}$$

여기서 $a_0, a_1, \cdots, a_k$는 다음 식들을 만족한다.

$$-\infty = a_0 < a_1 < \cdots < a_{k-1} < a_k = \infty \tag{3.4.26}$$

다음과 같은 확률들을 정의하자.

$$p(i) = Pr(x \in [a_{i-1}, a(i)]) = \int_{a_{i-1}}^{a(i)} f(x)dx, \;\; (i = 1, 2, \cdots, k) \tag{3.4.27}$$

소구간 $[a_{i-1}, a(i)]$에서 확률밀도함수 $f(x)$를 따르는 $n(i)$개 샘플들 $x_1^{(i)}, x_2^{(i)}, \cdots, x_{n(i)}^{(i)}$를 생성하고, 이 소구간에서 θ를 다음과 같이 추정하자.

$$\hat{\theta}(i) = \frac{1}{n(i)} \sum_{j=1}^{n(i)} g\left(x(j)^{(i)}\right), \;\; (i = 1, 2, \cdots, k) \tag{3.4.28}$$

식 (3.4.25)와 식 (3.4.28)에서 알 수 있듯이, 전구간에서 θ를 다음과 같이 추정할 수 있다.

$$\hat{\theta}_S = \sum_{i=1}^{k} p(i)\hat{\theta}(i) \tag{3.4.29}$$

이 $\hat{\theta}_S$를 층화추출법에 의한 추정값이라 한다. 당연한 말이지만, 각 소구간 내에서 함수 $g(x)$가 상수에 가까울수록 $\hat{\theta}_S$의 효율이 좋아진다.

예제 3.4.15 지지대가 $(0,1)$인 일양확률변수 u에 대해서 모수 $\theta \doteq E(g(u)) = \int_0^1 g(u)du$를 추정하기로 하자.

첫째, 원시몬테카를로법으로 θ를 추정하기 위해서는 먼저 지지대가 $(0,1)$인 일양샘플들 $u_1, u_2, \cdots, u_{2n-1}, u_{2n}$을 생성한 다음, θ를 다음과 같이 추정한다.

$$\hat{\theta}_C \doteq \frac{1}{2n}\sum_{j=1}^{2n} g\left(u(j)\right) \tag{1}$$

다음 식들이 성립함은 명백하다.

$$E\left(\hat{\theta}_C\right) = E\left(g\left(u\right)\right) = \theta \tag{2}$$

$$Var\left(\hat{\theta}_C\right) = \frac{1}{2n}Var\left(g\left(u\right)\right) = \frac{1}{2n}\left\{\int_0^1 g^2(u)du - \left[\int_0^1 g(u)du\right]^2\right\} \tag{3}$$

둘째, 층화추출법으로 θ를 추정하기 위해서, 다음 식을 살펴보자.

$$\theta = \int_0^{1/2} g(u)du + \int_{1/2}^1 g(u)du \tag{4}$$

지지대가 $(0,1)$인 일양샘플들 $u_1, u_2, \cdots, u_{2n-1}, u_{2n}$을 생성한 다음, θ를 다음과 같이 추정하자.

$$\hat{\theta}_S \doteq \frac{1}{2}\left[\frac{1}{n}\sum_{j=1}^{n} g\left(\frac{u(j)}{2}\right) + \frac{1}{n}\sum_{j=n+1}^{2n} g\left(\frac{u(j)+1}{2}\right)\right] \tag{5}$$

다음 상수들을 정의하자.

$$\theta_1 \doteq \int_0^{1/2} g(u)du, \quad \theta_2 \doteq \int_{1/2}^1 g(u)du \tag{6}$$

다음 식들이 성립한다.

$$E\left(\hat{\theta}_S\right) = \theta_1 + \theta_2 = \theta \tag{7}$$

$$Var\left(g\left(\frac{u(j)}{2}\right)\right) = 2\int_0^{1/2} g^2(u)du - 4\theta_1^2 \tag{8}$$

$$Var\left(g\left(\frac{u(j)+1}{2}\right)\right) = 2\int_{1/2}^{1} g^2(u)du - 4\theta_2^2 \tag{9}$$

식 (5), 식 (8), 그리고 식 (9)에서 알 수 있듯이, 다음 식이 성립한다.

$$Var\left(\hat{\theta}_S\right) = \frac{1}{2n}\left\{\int_0^1 g^2(u)du - 2\left[\theta_1^2 + \theta_2^2\right]\right\} \tag{10}$$

식 (3)과 식 (10)에서 알 수 있듯이, 다음 식이 성립한다.

$$Var\left(\hat{\theta}_S\right) = Var\left(\hat{\theta}_C\right) - \frac{1}{2n}\left[\theta_1 - \theta_2\right]^2 \tag{11}$$

따라서, 다음 식이 성립한다.

$$Var\left(\hat{\theta}_S\right) \le Var\left(\hat{\theta}_C\right) \tag{12}$$

즉, 층화추출법에 의한 추정값이 원시몬테카를로법에 의한 추정값보다 효율적이라고 할 수 있다. 특히, 소그룹의 평균들 θ_1과 θ_2의 차이가 클수록 층화추출법의 효율성이 좋아진다.

한 예로서, 정적분 $\theta = \int_0^1 e^u du$를 추정하기로 하자. 지지대가 $(0,1)$인 일양샘플들 $u_1, u_2, \cdots, u_{2n-1}, u_{2n}$에 대해서 원시몬테카를로법과 층화추출법에 의한 추정값들은 각각 다음과 같다.

$$\hat{\theta}_C = \frac{1}{2n}\sum_{j=1}^{2n} \exp\left(u(j)\right) \tag{13}$$

$$\hat{\theta}_S = \frac{1}{n}\sum_{j=1}^{n} \frac{\exp\left(\frac{u(j)}{2}\right) + \exp\left(\frac{u_{j+n}+1}{2}\right)}{2} \tag{14}$$

또한, 다음 식들이 성립한다.

$$\int_0^1 e^u du = e - 1 \tag{15}$$

$$\int_0^1 e^{2u} du - \left[\int_0^1 e^u du\right]^2 = \frac{e^2-1}{2} - [e^1 - 1]^2 = 2e^1 - \frac{1}{2}e^2 - \frac{3}{2} \approx 0.2420 \tag{16}$$

$$\theta_1 = \int_0^{1/2} e^u du = e^{1/2} - 1 \approx 0.6487 \tag{17}$$

$$\theta_2 = \int_{1/2}^1 e^u du = e - e^{1/2} \approx 1.0696 \tag{18}$$

따라서, 다음 식들이 성립한다.

$$Var\left(\hat{\theta}_C\right) = \frac{1}{2n}\left\{\int_0^1 e^{2u} du - \left[\int_0^1 e^u du\right]^2\right\} \approx \frac{0.1210}{n} \tag{19}$$

$$Var\left(\hat{\theta}_S\right) = Var\left(\hat{\theta}_C\right) - \frac{1}{2n}\left[2e^{1/2} - 1 - e\right]^2 \approx \frac{0.1210}{n} - \frac{0.0886}{n} = \frac{0.0325}{n} \tag{20}$$

따라서 원시몬테카를로법에 의한 추정값 $\hat{\theta}_C$에 대한 층화추출법에 의한 추정값 $\hat{\theta}_S$의 오차분산비는 $0.0325/0.1210 = 0.2686$이다.

원시몬테카를로법과 층화추출법을 비교하기 위해서, 다음 MATLAB프로그램 CrudeStratified101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
% Filename: Crude_Stratified101.m
% Crude vs Stratified Sampling Monte Carlo Integration
% Programmed by CBS
% ------------------------------------------------------------------------
clear all, close all
rand('twister',5489);   % Use twister or state for seed.
% Monte Carlo integration
Ns = 10000              % Number of Samples
u = rand(Ns,1);
for ndum = 1:1:40
    Ndum(ndum) = ndum*100;
    N1 = 1:Ndum(ndum);
    N2 = N1+Ns/2;
    dumMC01 = exp(u(N1));
    dumMC02 = exp(u(N2));
    dumMC = [ dumMC01; dumMC02 ];
    CrudeMC(ndum) = mean(dumMC);
    CrudeStd(ndum) = std(dumMC)/sqrt(2*Ndum(ndum));
    dumMC11 = exp(u(N1)/2)';
    dumMC12 = exp((u(N2)+1)/2)';
    dumMC1 = (dumMC11+dumMC12)/2;
    StratifiedMC(ndum) = mean(dumMC1);
    StratifiedStd(ndum) = std(dumMC1)/sqrt(Ndum(ndum));
    VarRatio1(ndum) = StratifiedStd(ndum)^2/CrudeStd(ndum)^2;
end
% Ploting
```

```
TrueInt = exp(1)-1
Ndum = 2*Ndum;
TI = TrueInt*ones(1,40);
subplot(2,1,1)
plot(Ndum,CrudeMC,'r-',Ndum,StratifiedMC,'g--','LineWidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[1.68,1.74])
legend('Crude','Stratified Sampling','location','SE')
hold on
plot([1 Ndum(end)],[TrueInt TrueInt],'k:','LineWidth',2)
xlabel('Number of Random Numbers','Fontsize',12');
ylabel('MC Integral','Fontsize',12');
hold off
subplot(2,1,2)
plot(Ndum,VarRatio1,'g--','LineWidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[0.25,0.30])
hold on
TrueVR = 0.0325/0.1210
plot([1 Ndum(end)],[TrueVR TrueVR],'k:','LineWidth',2)
xlabel('Number of Random Numbers','Fontsize',12');
ylabel('Variance Ratio','Fontsize',12');
hold off
saveas(gcf,'Crude_Stratified101','epsc')
save('Crude_Stratified101','CrudeMC','StratifiedMC')
% End of program
% --------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 정적분 $\theta = \int_0^1 e^u du$를 원시몬테카를로법과 층화추출법으로 추정한다. 즉, 일양샘플들의 개수를 200에서 8000까지 증가시켜가면서, 모수 θ를 원시몬테카를로법과 층화추출법으로 추정한다. 이 추정값들이 그림 **??**에 그려져 있다. 그림 **??**의 상단 그래프에서 적색 실선은 원시몬테카를로법에 의한 추정값 $\hat{\theta}_C$를, 그리고 녹색 긴점선은 층화추출법에 의한 추정값 $\hat{\theta}_S$을 나타낸다. 이 그래프에서 알 수 있듯이, 원시몬테카를로법에 의한 추정값보다 층화추출법에 의한 추정값이 진짜 정적분값에 빨리 수렴한다. 그림 **??**의 하단 그래프에서 녹색 긴점선은 원시몬테카를로법에 의한 추정값 $\hat{\theta}_C$에 대한 층화추출법에 의한 추정값 $\hat{\theta}_S$의 오차분산비를 나타낸다. 이 그래프에서 알 수 있듯이, 이 예제에서는 층화추출법이 원시몬테카르로법에 비해 훨씬 효율적이다. ■

예제 3.4.16 R언어를 사용해서 예제 3.4.15의 문제를 풀기 위해서, 다음 R프로그램 다음 R프로그램 CrudeStratified101R.R을 실행해 보자.

```
# --------------------------------------------------------------------------
# Filename: CrudeStratified101R.R
# Crude vs Stratified Monte Carlo 1
# Programmed by CBS
# --------------------------------------------------------------------------
# Simulation
set.seed(11)
Ns <- 10000               # Number of Samples
```

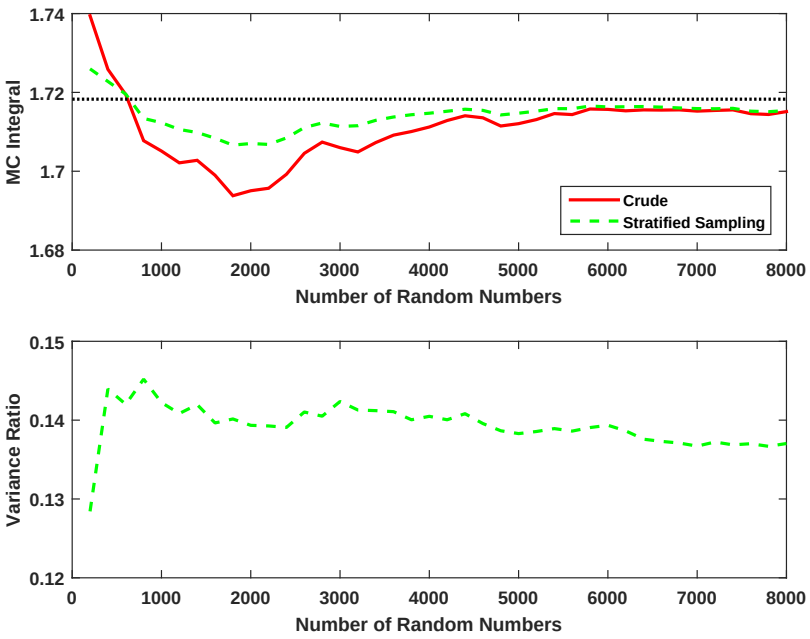


그림 3.4.13. 원시몬테카를로법과 층화추출법 1 (MATLAB)

```
u <- runif(Ns)
NoSim <- 40
Ndum <- rep(0,NoSim)
CrudeMC <- CrudeSd  <- rep(0,NoSim)
StratifiedMC <- StratifiedSd  <- rep(0,NoSim)
VarRatio <- rep(0,NoSim)

for( ndum in 1:1:NoSim ){
    Ndum[ndum] <- ndum*100
    N1 <- c(1:Ndum[ndum])
    N2 <- N1 + Ns/2
    dumMC01 <- exp(u[N1])
    dumMC02 <- exp(u[N2])
    dumMC <- rbind(dumMC01,dumMC02)
    CrudeMC[ndum] <- mean(dumMC)
    CrudeSd[ndum] <- sd(dumMC)/sqrt(2*Ndum[ndum])
    dumMC11 <- exp(u[N1]/2)
    dumMC12 <- exp((u[N2]+1)/2)
    dumMC1 <- (dumMC11+dumMC12)/2
    StratifiedMC[ndum] <- mean(dumMC1)
    StratifiedSd[ndum] <- sd(dumMC1)/sqrt(Ndum[ndum])
    VarRatio[ndum] <- StratifiedSd[ndum]^2/CrudeSd[ndum]^2
}
VarRatio

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('CrudeStratified101R.eps')  # Start to save figure
RVdata <- data.frame(CrudeMC,StratifiedMC,VarRatio)
thetaa <- exp(1)-1
plotMC <- ggplot(RVdata,aes(x=Ndum,y=CrudeMC)) +
```

```
          geom_point(shape=16,col="red",lwd=1.7) +
          geom_line(col="red",linetype=1) +
          geom_point(aes(x=Ndum,y=StratifiedMC),shape=16,col="green",lwd=1.7)+
          geom_line(aes(x=Ndum,y=StratifiedMC),col="green",linetype=2) +
          geom_hline(yintercept=thetaa,col="dark blue",lwd=1) +
          ylim(1.68,1.74) +
          xlab("Number of Samples") + ylab("MC estimate")
trueVR <- 0.2686
plotVR <- ggplot(RVdata,aes(x=Ndum,y=VarRatio)) +
          geom_point(shape=16,col="green",lwd=1.7) +
          geom_line(col="green",linetype=2) +
          geom_hline(yintercept=trueVR,col="dark blue",lwd=1) +
          ylim(0.25,0.30) +
          xlab("Number of Samples") + ylab("Variance Ratio")
pushViewport(viewport(layout=grid.layout(2,1)))
print(plotMC, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plotVR, vp=viewport(layout.pos.row=2, layout.pos.col=1))
dev.off()            # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 정적분 $\theta = \int 0^1 e^u du$를 원시몬테카를로법과 층화추출법으로 추정한다. 즉, 일양샘플들의 개수를 200에서 8000까지 증가시켜가면서, 모수 θ를 원시몬테카를로법과 층화추출법으로 추정한다. 이 추정값들이 그림 3.4.14에 그려져 있다. 그림 3.4.14의 상단 그래프에서 적색 실선은 원시몬테카를로법에 의한 추정값 $\hat{\theta}C$를, 그리고 녹색 긴점선은 층화추출법에 의한 추정값 $\hat{\theta}S$을 나타낸다. 이 그래프에서 알 수 있듯이, 원시몬테카를로법에 의한 추정값보다 층화추출법에 의한 추정값이 진짜 정적분값에 빨리 수렴한다. 그림 3.4.14의 하단 그래프에서 녹색 긴점선은 원시몬테카를로법에 의한 추정값 $\hat{\theta}C$에 대한 층화추출법에 의한 추정값 $\hat{\theta}S$의 오차분산비를 나타낸다. 이 그래프에서 알 수 있듯이, 이 예제에서는 층화추출법이 원시몬테카르로법에 비해 훨씬 효율적이다. ■

다음 정적분을 구하기로 하자.

$$\theta = \int_0^1 g(x)dx \tag{3.4.30}$$

이 정적분을 다음과 같이 쓸 수 있다.

$$\theta = \sum_{i=1}^{k} \int_{[i-1]/k}^{i/k} g(x)dx \tag{3.4.31}$$

여기서 $\theta(i)$는 다음과 같다.

$$\theta(i) \doteq \int_{[i-1]/k}^{i/k} g(x)dx \tag{3.4.32}$$

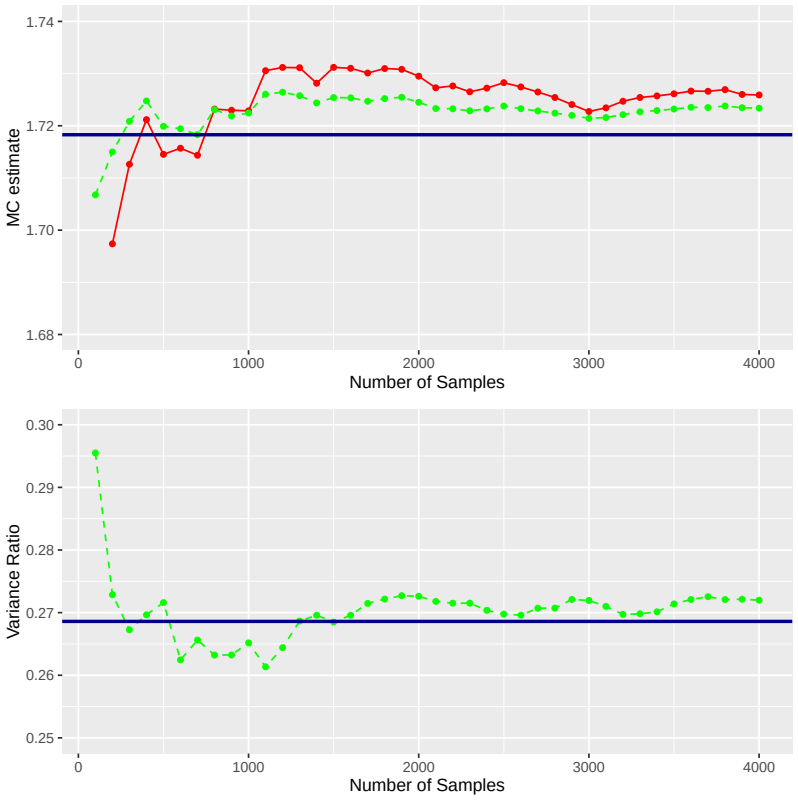


그림 3.4.14. 원시몬테카를로법과 층화추출법 1 (R)

지지대 $(0,1)$인 일양확률분포에서 생성된 일양샘플들 $u_{i,1}, u_{i,2}, \cdots, u_{i,n(i)}$을 사용해서 정적분 $\theta(i)$를 다음과 같이 추정하자.

$$\hat{\theta}(i) = \frac{1}{n(i)} \sum_{j=1}^{n(i)} g\left(\frac{u_{i,j}+i-1}{k}\right) \tag{3.4.33}$$

식 (3.4.25)와 식 (3.4.28)에서 알 수 있듯이, 전구간에서 θ를 다음과 같이 추정할 수 있다.

$$\hat{\theta}_S = \frac{1}{n} \sum_{i=1}^{k} n(i)\hat{\theta}(i) \tag{3.4.34}$$

여기서 $n = \sum_{i=1}^{k} n(i)$이다. 식 (3.4.34)를 사용하는데 유의할 점은 추정값들 $\hat{\theta}_1, \hat{\theta}_2, \cdots, \hat{\theta}_k$는 서로 독립이 아니라는 것이다. 이러한 비독립성은 $\{u_{i,j}\}$가 아닌 $\{u_{i,j}+i\}$에서 오는 것이다. 따라서, 다음 식이 성립하지 않는다.

$$Var\left(\hat{\theta}_S\right) = \frac{1}{n^2} \sum_{i=1}^{k} n(i)^2 Var\left(\hat{\theta}(i)\right) \tag{3.4.35}$$

식 (3.4.35) 대신에 다음 식을 사용해야 한다.

$$Var\left(\hat{\theta}_S\right) = \frac{1}{n^2}\sum_{i=1}^{k}\sum_{j=1}^{k} n(i)n(j)Cov\left(\hat{\theta}_i,\hat{\theta}(j)\right) \tag{3.4.36}$$

결과적으로, 층화추출법에 의한 추정량 $\hat{\theta}_S$의 분산이 반드시 원시몬테카를로법에 의한 추정량 $\hat{\theta}_C$의 분산과 큰 차이가 있다고 말할 수는 없다.

예제 3.4.17 다음 모수 θ를 추정하기로 하자.

$$\theta \doteq \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x^2\right) dx \tag{1}$$

식 $\theta = 0$이 성립함은 명백하다.

첫째, 원시몬테카를로법으로 θ를 추정하기 위해서는 먼저 지지대가 $(0,1)$인 일양샘플들 $u_1, u_2, \cdots, u_n$을 생성한 다음, θ를 다음과 같이 추정한다.

$$\hat{\theta}_C \doteq \frac{1}{n}\sum_{j=1}^{n} N^{-1}\left(u(j)\right) \tag{2}$$

다음 식들이 성립함은 명백하다.

$$E\left(\hat{\theta}_C\right) = \theta = 0, \quad Var\left(\hat{\theta}_C\right) = \frac{1}{n} \tag{3}$$

둘째, 다음과 같이 $\hat{\theta}(i)$를 추정하자.

$$\hat{\theta}(i) = \frac{1}{n/k}\sum_{j=[i-1]n/k+1}^{in/k} N^{-1}\left(\frac{u(j)+i-1}{k}\right) \tag{4}$$

층화추출법에 의한 θ의 추정값은 다음과 같다.

$$\hat{\theta}_S = \frac{1}{k}\sum_{i=1}^{k}\hat{\theta}(i) \tag{5}$$

앞에서도 언급했듯이, $\hat{\theta}_1, \hat{\theta}_2, \cdots, \hat{\theta}_k$가 서로 독립이 아니므로, 층화추출법에 의한 추정량 $\hat{\theta}_S$의 분산이 원시몬테카를로법에 의한 추정량 $\hat{\theta}_C$의 분산과 큰 차이가 있다고 말할 수는 없다.

원시몬테카를로법과 층화추출법을 비교하기 위해서, 다음 MATLAB프로그램 CrudeStratified102.m을 실행해 보자.

```
% ------------------------------------------------------------------------
% Filename: CrudeStratified102.m
% Crude vs Stratified Sampling Monte Carlo Integration 2
% Programmed by CBS
% ------------------------------------------------------------------------
clear all, close all
rand('twister',5489);   % Use twister or state for seed.
% Monte Carlo integration
Ns = 1000               % Number of Samples
Sn = 10                 % Number of Strata
ni = Ns/Sn              % Number of samples in a stratum
u = rand(Ns,1);
% Crude Monte Carlo
Crude = icdf('normal',u,0,1);
MeanCrude = mean(Crude)
VarCrude = var(Crude)
xx = -3.5:0.25:3.5;
histCrude = hist(Crude,xx);
subplot(2,2,1)
bar(xx,histCrude,'w');
set(gca,'fontsize',11,'fontweigh','bold','xlim',[-3.7,3.7])
subplot(2,2,2)
autocorr(Crude,[],2)
% Stratified Sampling Monte Carlo
for jj=1:Sn
    v(((jj-1)*ni+1):jj*ni) = (u(((jj-1)*ni+1:jj*ni))+(jj-1))/Sn;
end
SS = icdf('normal',v,0,1);
MeanSS = mean(SS)
VarSS = var(SS)
histSS = hist(SS,xx);
subplot(2,2,3)
bar(xx,histSS,'w');
set(gca,'fontsize',11,'fontweigh','bold','xlim',[-3.7,3.7])
subplot(2,2,4)
autocorr(SS,[],2)
saveas(gcf,'CrudeStratified102','epsc')
save('CrudeStratified102','Crude','SS')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB 프로그램을 실행하면, 정적분 θ를 원시몬테카를로법과 층화추출법으로 추정한다. 지지대가 $(0,1)$인 일양샘플들 $u_1, u_2, \cdots, u_{1000}$에 역함수법을 적용해서 구한 정규샘플들 $N^{-1}(u_1), N^{-1}(u_2), \cdots, N^{-1}(u_{1000})$들을 사용한 원시몬테카를로법에 의한 추정값은 $\hat{\theta}_C = -0.0342$이고 표본분산은 0.9125이다. 또한, 식 (5)를 사용한 층화추출법에 의한 추정값은 $\hat{\theta}_S = -0.0026$이고 표본분산은 0.9787이다. 따라서, 층화추출법에 의한 추정값 $\hat{\theta}_S = -0.0026$가 원시몬테카를로법에 의한 추정값 $\hat{\theta}_C = -0.0342$보다 모수에 가까우나, 층화추출법에 의한 표본분산이 원시몬테카를로법에 의한 표본분산보다 크다.

원시몬테카를로법에서 사용한 정규샘플들 $\{N^{-1}(u_1)\}$의 히스토그램과 자기상관함수(autocorrelation funtion), 층화추출법에서 사용한 정규샘플들 $\left\{N^{-1}\left(\frac{uj+i-1}{k}\right)\right\}$의 히스토그램과 자기상관함수가 그림 3.4.15에 그려져 있다. 그림 3.4.15의 좌측상단에는 원시몬테카를로

법에 사용된 정규샘플들의 히스토그램이 그려져 있고, 우측상단에는 이 정규샘플들의 자기상관함수가 그려져 있다. 또한, 좌측하단에는 층화추출법에 사용된 정규샘플들의 히스토그램이 그려져 있고, 우측하단에는 이 정규샘플들의 자기상관함수가 그려져 있다. 이 그래프들에서 알 수 있듯이, 층화추출법에 사용된 정규샘플들의 히스토그램이 원시몬테카를로법에 사용된 정규샘플들의 히스토그램보다 더 정규확률분포에 가깝다. 우측상단의 자기상관함수에서 알 수 있듯이, 원시몬테카를로법에 사용된 정규샘플들은 서로 독립이라고 할 수 있다. 그러나, 우측하단의 자기상관함수에서 알 수 있듯이, 층화추출법에 사용된 정규샘플들 사이에는 높은 자기상관관계가 성립한다. 이러한 이유로 층화추출법에 의한 분산이 원시몬테카를로법에 의한 분산보다 크다. ■

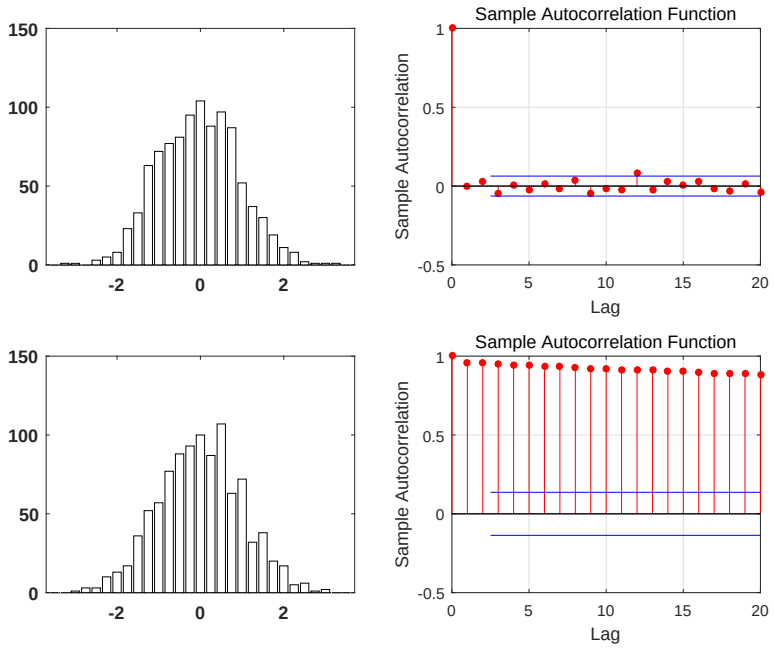


그림 3.4.15. 원시몬테카를로법과 층화추출법 2 (MATLAB)

예제 3.4.18 R언어를 사용해서 예제 3.4.17의 문제를 풀기 위해서, 다음 R프로그램 다음 R프로그램 CrudeStratified102R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: CrudeStratified102R.R
# Crude vs Stratified Monte Carlo 2
# Programmed by CBS
# ------------------------------------------------------------------------
# Simulation
set.seed(11)
Ns <- 1000              # Number of Samples
Sn <- 10                # Number of Strata
```

```
ni <- Ns/Sn             # Number of samples in a stratum
u <- runif(Ns)

# Crude Monte Carlo
Crude <- qnorm(u, mean=0, sd=1, lower.tail=TRUE, log.p=FALSE)
( MeanCrude <- mean(Crude) )
( VarCrude <- var(Crude) )
## ACF
conf.level <- 0.95
CrudeCIline <- qnorm((1 - conf.level)/2)/sqrt(length(Crude))
CrudeACF <- acf(Crude, plot=FALSE)
CrudeACFdf <- with(CrudeACF, data.frame(lag,acf))

# Stratified Sampling Monte Carlo
v <- rep(0,Ns)
for (jj in 1:Sn){
    for (ii in 1:ni){
        kk <- (jj-1)*ni + ii
        v[kk] <- ( u[kk]+(jj-1) )/Sn
    }
}
SS <- qnorm(v)
( MeanSS <- mean(SS) )
( VarSS <- var(SS) )
## ACF
conf.level <- 0.95
SSCIline <- qnorm((1 - conf.level)/2)/sqrt(length(SS))
SSACF <- acf(SS, plot=FALSE)
SSACFdf <- with(SSACF, data.frame(lag,acf))

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('CrudeStratified102R.eps')  # Start to save figure
RVdata <- data.frame(Crude,SS)
plot11 <- ggplot(RVdata,aes(x=Crude)) +
          geom_histogram(binwidth=0.25,fill="white",col="red") +
          xlim(-3,3)
plot12 <- ggplot(data=CrudeACFdf, mapping=aes(x=lag, y=acf)) +
          geom_bar(stat = "identity", position = "identity") +
          geom_hline(yintercept=CrudeCIline, col="red", linetype=2) +
          geom_hline(yintercept=-CrudeCIline, col="red", linetype=2)
plot21 <- ggplot(RVdata,aes(x=SS)) +
          geom_histogram(binwidth=0.25,fill="white",col="dark blue") +
          xlab("Stratified") +
          xlim(-3,3)
plot22 <- ggplot(data=SSACFdf, mapping=aes(x=lag, y=acf)) +
          geom_bar(stat = "identity", position = "identity") +
          geom_hline(yintercept=SSCIline, col="dark blue", linetype=2) +
          geom_hline(yintercept=-SSCIline, col="dark blue", linetype=2)
pushViewport(viewport(layout=grid.layout(2,2)))
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot21, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot22, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()             # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 정적분 θ를 원시몬테카를로법과 층화추출법으로 추정한다. 지지대가 $(0,1)$인 일양샘플들 $u_1, u_2, \cdots, u_{1000}$에 역함수법을 적용해서 구한 정규샘플들 $N^{-1}(u_1), N^{-1}(u_2), \cdots, N^{-1}(u_{1000})$들을 사용한 원시몬테카를로법에 의한 추정값은 $\hat{\theta}_C = -0.01598$이고 표본분산은 0.98955이다. 또한, 층화추출법에 의한 추정값은 $\hat{\theta}_S = -0.01118$이고 표본분산은 1.027523이다.

원시몬테카를로법에서 사용한 정규샘플들 $\left\{N^{-1}(u_1)\right\}$의 히스토그램과 자기상관함수(autocorrelation funtion), 층화추출법에서 사용한 정규샘플들 $\left\{N^{-1}\left(\frac{uj+i-1}{k}\right)\right\}$의 히스토그램과 자기상관함수가 그림 3.4.16에 그려져 있다. 그림 3.4.16의 좌측상단에는 원시몬테카를로법에 사용된 정규샘플들의 히스토그램이 그려져 있고, 우측상단에는 이 정규샘플들의 자기상관함수가 그려져 있다. 또한, 좌측하단에는 층화추출법에 사용된 정규샘플들의 히스토그램이 그려져 있고, 우측하단에는 이 정규샘플들의 자기상관함수가 그려져 있다. 이 그래프들에서 알 수 있듯이, 층화추출법에 사용된 정규샘플들의 히스토그램이 원시몬테카를로법에 사용된 정규샘플들의 히스토그램보다 더 정규확률분포에 가깝다. 우측상단의 자기상관함수에서 알 수 있듯이, 원시몬테카를로법에 사용된 정규샘플들은 서로 독립이라고 할 수 있다. 그러나, 우측하단의 자기상관함수에서 알 수 있듯이, 층화추출법에 사용된 정규샘플들 사이에는 높은 자기상관관계가 성립한다. 이러한 이유로 이 층화추출법에 의한 분산이 원시몬테카를로법에 의한 분산보다 크다. ■

3.4.5 요점추출법

다음 정적분을 구하기로 하자.

$$\theta = E_f(g(x)) = \int_{-\infty}^{\infty} g(x)f(x)dx \tag{3.4.37}$$

여기서 기대값연산자 E_f는 확률밀도함수 $f(x)$에 대한 기대값을 의미한다. 함수 $g(x)$와 정의역이 동일한 어떤 확률밀도함수 $h(x)$에 대해서 다음 식들이 성립한다.

$$\theta = \int_{-\infty}^{\infty} \frac{g(x)f(x)}{h(x)}h(x)dx = E_h\left(\frac{g(x)f(x)}{h(x)}\right) \tag{3.4.38}$$

만약 함수 $g(x)f(x)/h(x)$가 상수에 가깝도록 확률밀도함수 $h(x)$를 선택할 수 있고 또한 확률밀도함수 $h(x)$를 따르는 샘플들의 생성이 쉽다면, 다음 식을 사용해서 정적분 θ를 추정할

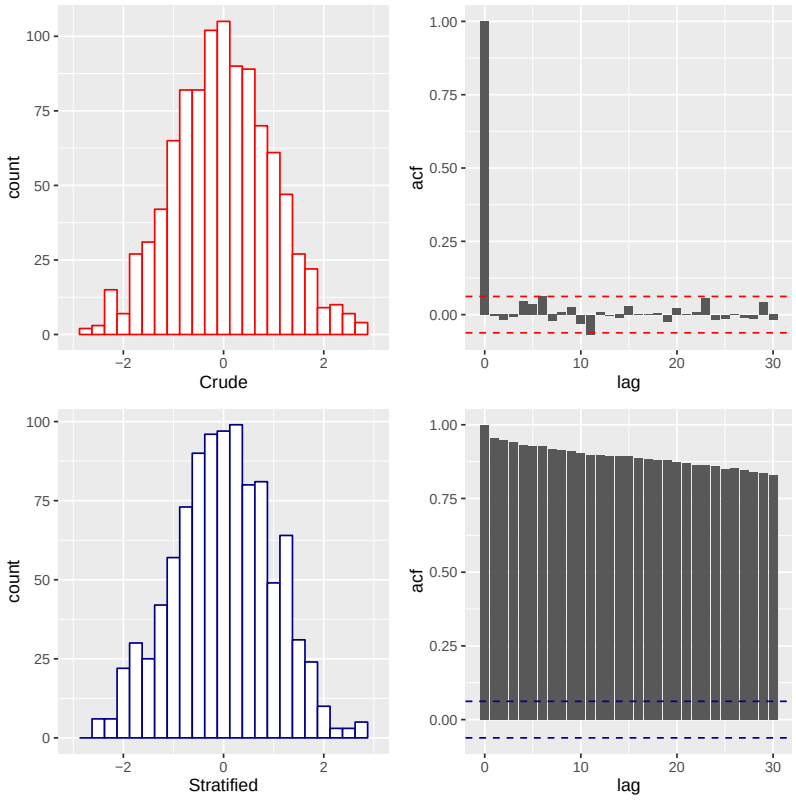


그림 3.4.16. 원시몬테카를로법과 층화추출법 2 (R)

수 있다.

$$\hat{\theta} = \frac{1}{n}\sum_{j=1}^{n} \frac{g(x(i))f(x(i))}{h(x(i))} \tag{3.4.39}$$

여기서 $x_1, x_2, \cdots, x_n$는 확률밀도함수 $h(x)$에서 생성된 샘플들이다.

예제 3.4.19 원시몬테카를로법과 요점추출법을 사용해서 다음 확률 θ를 추정하기로 하자.

$$\theta(c) \doteq \int_c^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x^2\right) dx = 1 - \mathcal{N}(c) = \mathcal{N}(-c) \tag{1}$$

여기서 $\mathcal{N}(\cdot)$는 표준정규확률분포의 누적확률분포함수이다. 다음 식이 성립한다.

$$\theta(c) = \int_{-\infty}^{\infty} 1(c < x) n\left(x \mid 0, 1^2\right) dx \tag{2}$$

첫째, 원시몬테카를로법으로 $\theta(c)$를 추정하기 위해서는 먼저 표준정규샘플들 $x_1, x_2, \cdots, x_n$

을 생성한 다음, $\theta(c)$를 다음과 같이 추정한다.

$$\hat{\theta}_C(c) = \frac{1}{n}\sum_{j=1}^{n} 1\left(x(j) > c\right) \tag{3}$$

이항모형에서 알 수 있듯이, 확률변수 $\sum_{j=1}^{n} 1\left(x(j) > c\right)$는 모수들이 $(n, \theta(c))$인 이항확률분포를 따른다. 따라서, 추정량 $\hat{\theta}_C(c)$의 평균과 분산은 각각 다음과 같다.

$$E\left(\hat{\theta}_C(c)\right) = \theta(c), \quad Var\left(\hat{\theta}_C(c)\right) = \frac{1}{n}\theta(c)\left[1-\theta(c)\right] \tag{4}$$

둘째, 요점추출법을 사용해서 $\theta(c)$를 추정하기로 하자. 다음 식들이 성립한다.

$$\begin{aligned}
\theta(c) &= \int_{-\infty}^{\infty} 1(c<x) n\left(x \mid 0, 1^2\right) dx \\
&= \int_{-\infty}^{\infty} 1(c<x) \frac{n\left(x \mid 0, 1^2\right)}{n\left(x \mid c, 1^2\right)} n\left(x \mid c, 1^2\right) dx \\
&= \int_{-\infty}^{\infty} 1(c<x) \exp\left(-cx + \frac{1}{2}c^2\right) n\left(x \mid c, 1^2\right) dx
\end{aligned} \tag{5}$$

식 (5)에서 알 수 있듯이, 요점추출법에 의한 $\theta(c)$의 추정량은 다음과 같다.

$$\hat{\theta}_{IP}(c) \doteq \exp\left(\frac{1}{2}c^2\right)\frac{1}{n}\sum_{j=1}^{n} 1\left(x(j) > c\right)\exp\left(-cx(j)\right) \tag{6}$$

여기서 $x_1, x_2, \cdots, x_n$은 정규확률분포 $N(c,1)$을 따르는 정규샘플들이다. 따라서, $\{z_j \doteq x_j - c \mid j = 1, 2, \cdots, n\}$은 표준정규샘플들이다. 식 (6)에서 알 수 있듯이, 다음 식이 성립한다.

$$\hat{\theta}_{IP}(c) \doteq \exp\left(\frac{1}{2}c^2\right)\frac{1}{n}\sum_{j=1}^{n} 1\left(z(j) > 0\right)\exp\left(-c\left[z(j)+c\right]\right) \tag{7}$$

다음 식이 성립함은 명백하다.

$$E\left(\hat{\theta}_{IP}\right) = \theta(c) \tag{8}$$

다음 식들이 성립한다.

$$\begin{aligned} & E\left([1(x>c)\exp(-cx)]^2\right) \\ &= \int_c^\infty \exp(-2cx)\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}[x-c]^2\right)dx \\ &= \int_c^\infty \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}[x+c]^2\right)dx \\ &= \int_{2c}^\infty \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}z^2\right)dz \end{aligned} \tag{9}$$

즉, 다음 식들이 성립한다.

$$E\left([1(x>c)\exp(-cx)]^2\right) = 1-\mathcal{N}(2c) = \mathcal{N}(-2c) \tag{10}$$

식 (8)과 식 (10)에서 알 수 있듯이, 다음 식들이 성립한다.

$$Var\left(\exp\left(\frac{1}{2}c^2\right)1(x>c)\exp(-cx)\right) = \exp(c^2)N(-2c)-\theta^2(c) \tag{11}$$

즉, 다음 식이 성립한다.

$$Var\left(\hat{\theta}_{IS}\right) = \frac{1}{n}\left\{\exp(c^2)N(-2c)-\theta^2(c)\right\} \tag{12}$$

원시몬테카를로법과 요점추출법을 비교하기 위해서, 다음 MATLAB프로그램 CrudeImportSample101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
% Filename: CrudeImportSample101.m
% Crude vs Importantance Sampling 1
% Programmed by CBS
% ------------------------------------------------------------------------
clear all, close all
rand('twister',5489);   % Use twister or state for seed.
Ns = 3000               % Number of Samples
x = randn(Ns,1);
for cdum = 1:1:40
    cp(cdum) = 3.0+cdum/10;
    theta(cdum) = normcdf(-cp(cdum));
    dumMC = (x>cp(cdum));
    CrudeMC(cdum) = mean(dumMC);
    CrudeStd(cdum) = std(dumMC)/sqrt(Ns);
    dumMC2 = exp(cp(cdum)^2/2)*(x>0).*exp(-cp(cdum)*(x+cp(cdum)));
    ImportMC(cdum) = mean(dumMC2);
    ImportStd(cdum) = std(dumMC2)/sqrt(Ns);
    VarRatio(cdum) = ImportStd(cdum)^2/CrudeStd(cdum)^2;
end
```

```
% Ploting
subplot(1,2,1)
plot(cp,theta,'b:',cp,CrudeMC,'r-',cp,ImportMC,'g--','LineWidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
legend('True Value','Crude','Importance Sampling','location','NE')
axis([3 7 0 0.001])
xlabel('\it c','Fontsize',12','Fontweigh','bold');
ylabel('MC Integral','Fontsize',12','Fontweigh','bold');
subplot(1,2,2)
plot(cp,CrudeStd,'r-',cp,ImportStd,'g--','LineWidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[3 7])
xlabel('\it c','Fontsize',12','Fontweigh','bold');
ylabel('Standard Deviation','Fontsize',12');
hold off
saveas(gcf,'CrudeImportSample101','epsc')
save('CrudeImportSample101','CrudeMC','ImportMC')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB명령문이 실행하면, 각 시뮬레이션의 길이, 즉 일양샘플들의 개수를 3000으로 하고 또한 c를 3.1에서 7.0까지 증가시켜가며, 원시몬테카를로법과 요점추출법을 사용해서 $\theta(c)$를 추정한다. 이 추정값이 그림 3.4.17에 그려져 있다. 그림 3.4.17의 좌측 그래프에서 적색 실선은 원시몬테카를로법에 의한 추정값 $\hat{\theta}_C(c)$를 나타내고 녹색 긴점선은 요점추출법에 의한 추정값 $\hat{\theta}_{IP}(c)$를 나타낸다. 이 좌측 그래프에서 알 수 있듯이, 원시몬테카를로법에 의한 추정값보다 요점추출법에 의한 추정값이 진짜 정적분에 가깝다. 그림 3.4.17의 우측 그래프에서 적색 실선은 원시몬테카를로법에 의한 추정값 $\hat{\theta}_C(c)$의 표준오차를 나타내고 녹색 긴점선은 요점추출법에 의한 추정값 $\hat{\theta}_{IP}(c)$의 표준오차를 나타낸다. 이 우측 그래프에서 알 수 있듯이, 요점추출법에 의한 표준오차가 원시몬테카를로법에 의한 표준오차보다 훨씬 작다. 따라서, 요점추출법이 효율적이다. ■

예제 3.4.20 R언어를 사용해서 예제 3.4.19의 문제를 풀기 위해서, 다음 R프로그램 다음 R프로그램 CrudeImportSample101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: CrudeImportSample101R.R
# Crude vs Importance Sampling Monte Carlo
# Programmed by CBS
# ------------------------------------------------------------------------
# Simulation
set.seed(11)
Ns <- 3000
x <- rnorm(Ns)

NoSim <- 40
cp  <- thetaa <- rep(0,NoSim)
CrudeMC <- CrudeSd  <- rep(0,NoSim)
```

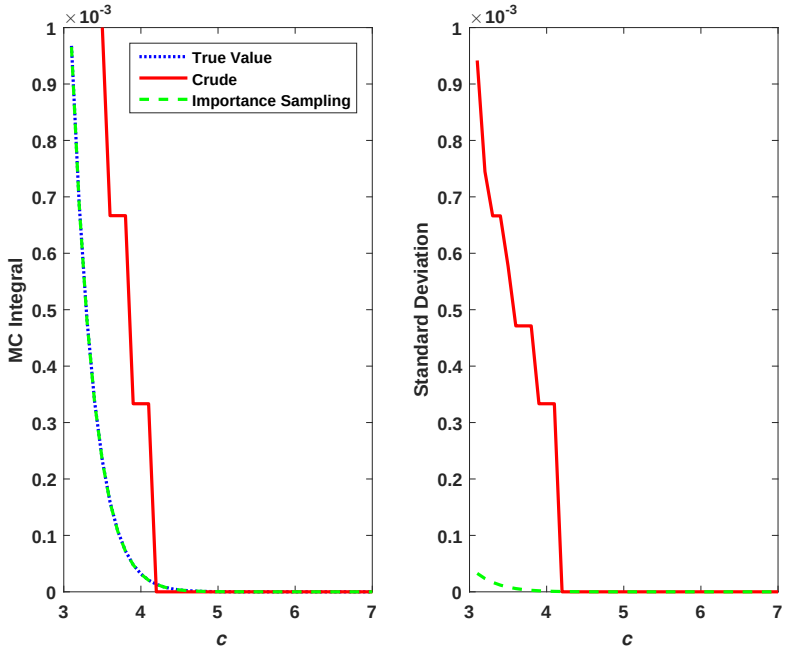


그림 3.4.17. 원시몬테카를로법과 요점추출법 (MATLAB)

```
ImportMC <- ImportSd  <- rep(0,NoSim)
VarRatio <- rep(0,NoSim)
for( cdum in 1:1:NoSim ){
    cp[cdum] <- 3.0 + cdum/10
    thetaa[cdum] <- pnorm(-cp[cdum])
    dumMC <- (x>cp[cdum])
    CrudeMC[cdum] <- mean(dumMC)
    CrudeSd[cdum] <- sd(dumMC)/sqrt(Ns)
    dumMC2 <- exp(cp[cdum]^2/2)*(x>0)*
              exp( -cp[cdum]*(x+cp[cdum]) )
    ImportMC[cdum] <- mean(dumMC2)
    ImportSd[cdum] <- sd(dumMC2)/sqrt(Ns)
    VarRatio[cdum] <- ImportSd[cdum]^2/CrudeSd[cdum]^2
}
VarRatio

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('CrudeImportSample101R.eps')  # Start to save figure
RVdata <- data.frame(cp,thetaa,CrudeMC,ImportMC,VarRatio)
plotMC <- ggplot(RVdata,aes(x=cp,y=thetaa)) +
          geom_line(col="black",linetype=3,lwd=1.2) +
          geom_line(aes(x=cp,y=CrudeMC),col="red",linetype=1,lwd=1) +
          geom_line(aes(x=cp,y=ImportMC),col="green",linetype=2,lwd=1) +
          ylim(0,0.001) +
          xlab("c") + ylab("MC estimate")
plotVR <- ggplot(RVdata) +
          geom_line(aes(x=cp,y=CrudeSd),col="red",linetype=1,lwd=1) +
          geom_line(aes(x=cp,y=ImportSd),col="green",linetype=2,lwd=1) +
      #    ylim(0,0.001) +
```

```
          xlab("c") + ylab("Standard Deviation")
pushViewport(viewport(layout=grid.layout(1,2)))
print(plotMC, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plotVR, vp=viewport(layout.pos.row=1, layout.pos.col=2))
dev.off()             # End to save figure
# ------------------------------------------------------------------------------
```

이 R명령문이 실행하면, 각 시뮬레이션의 길이, 즉 일양샘플들의 개수를 3000으로 하고 또한 c를 3.1에서 7.0까지 증가시켜가며, 원시몬테카를로법과 요점추출법을 사용해서 $\theta(c)$를 추정한다. 이 추정값이 그림 3.4.18에 그려져 있다. 그림 3.4.18의 좌측 그래프에서 적색 실선은 원시몬테카를로법에 의한 추정값 $\hat{\theta}_C(c)$를 나타내고 녹색 긴점선은 요점추출법에 의한 추정값 $\hat{\theta}_{IP}(c)$를 나타낸다. 이 좌측 그래프에서 알 수 있듯이, 원시몬테카를로법에 의한 추정값보다 요점추출법에 의한 추정값이 진짜 정적분에 가깝다. 그림 3.4.18의 우측 그래프에서 적색 실선은 원시몬테카를로법에 의한 추정값 $\hat{\theta}_C(c)$의 표준오차를 나타내고 녹색 긴점선은 요점추출법에 의한 추정값 $\hat{\theta}_{IP}(c)$의 표준오차를 나타낸다. 이 우측 그래프에서 알 수 있듯이, 요점추출법에 의한 표준오차가 원시몬테카를로법에 의한 표준오차보다 훨씬 작다. 따라서, 요점추출법이 효율적이다. ■

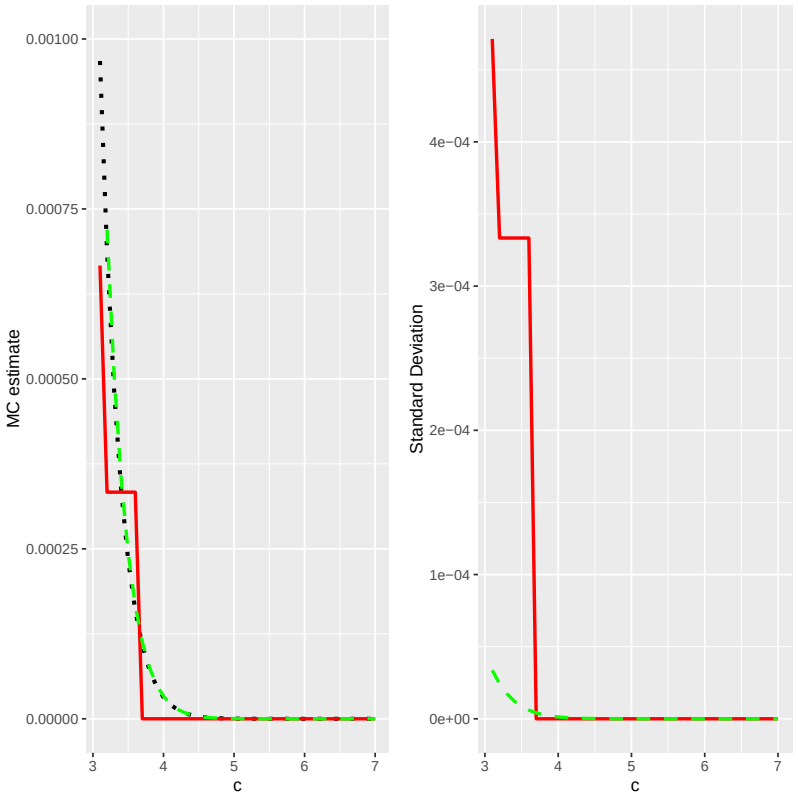


그림 3.4.18. 원시몬테카를로법과 요점추출법 (R)

예제 3.4.21 요점추출법을 사용해서 정규샘플들로부터 t 샘플들을 발생시키기로 하자. 다음 식이 성립한다.

$$t_k(x) = \frac{t_k(x)}{n(x)} n(x) \tag{1}$$

여기서 $t_k(x)$ 와 $n(x)$ 는 각각 자유도가 k 인 t 확률변수와 표준정규확률변수의 확률밀도함수들이다. 따라서, 다음 식이 성립한다.

$$t_k(x(j)) = w(j) n(x(j)), \quad (j = 1, 2, \cdots, n) \tag{2}$$

여기서 $w(j)$ 는 다음과 같이 정의되는 가중값이다.

$$w(j) \doteq t_k(x(j))/n(x(j)) \tag{3}$$

즉, 정규샘플들 $x_1, x_2, \cdots, x_n$ 에 가중값을 부여하면, t 샘플들 $t_1, t_2, \cdots, t_n$ 를 발생시킬 수 있다.

요점추출법을 사용해서 t 샘플들을 발생시키기 위해서, 다음 MATLAB프로그램 CrudeImportSample102.m을 실행해 보자.

```
% ------------------------------------------------------------------------
% Filename: CrudeImportSample102.m
% Generate t-random numbers using an Importance sampling
% Programmed by CBS
% ------------------------------------------------------------------------
clear all, close all
rand('twister',5489);          % Use twister or state for seed.
Ns = 20000;                    % Number of Samples
df = 5
zz = randn(0.5*Ns,1);
w = tpdf(zz,df)./normpdf(zz);
weight = w./sum(w);
tt = randsample(zz,Ns,true,weight);
% Histogram by an Importance Sampling
xx = -3.7:0.2:3.7;
[Nbar,xcenter] = hist(tt,xx);
Nheight = Nbar/(xcenter(2)-xcenter(1))/Ns;
bar(xcenter,Nheight,'white')
set(gca,'fontsize',11,'fontweigh','bold')
axis([ -4 4 0 0.5 ])
hold on
tTrue = pdf('t',xx,df);
normalTrue = normpdf(xx)
plot(xx,tTrue,'r-',xx,normalTrue,'b--','LineWidth',2)
legend('Importance Sampling','True PDF','Noraml PDF',1)
xlabel('\it x','Fontsize',12','Fontweigh','bold');
ylabel('PDF','Fontsize',12','Fontweigh','bold');
```

```
saveas(gcf,'CrudeImportSample102','epsc')
save('CrudeImportSample102','w','tt')
% End of program
% ------------------------------------------------------------------------
```

MATLAB함수 randsample은 주어진 확률분포에서 샘플을 추출하기 위한 것이다. 다음 MATLAB명령문을 실행해 보자.

```
>> tt = randsample(zz,Ns,true,weight)
```

이 명령문을 실행하면, 벡터 zz에 저장된 값들의 경험확률분포로부터 가중값 weight를 주며 샘플들을 샘플을 추출한다.

이 MATLAB프로그램을 실행하면, 10000개의 표준정규샘플들을 생성해서 만든 경험확률분포에서 식 (3)의 가중값을 주어 20000개 t샘플들을 생성한다. 이렇게 생성된 자유도가 5인 t샘플들의 히스토그램이 그림 3.4.19에 그려져 있다. 그림 3.4.19에서 백색 막대는 이 히스토그램이고 적색 실선은 자유도가 5인 t확률밀도함수이고, 청색 긴점선은 표준정규확률밀도함수이다. ■

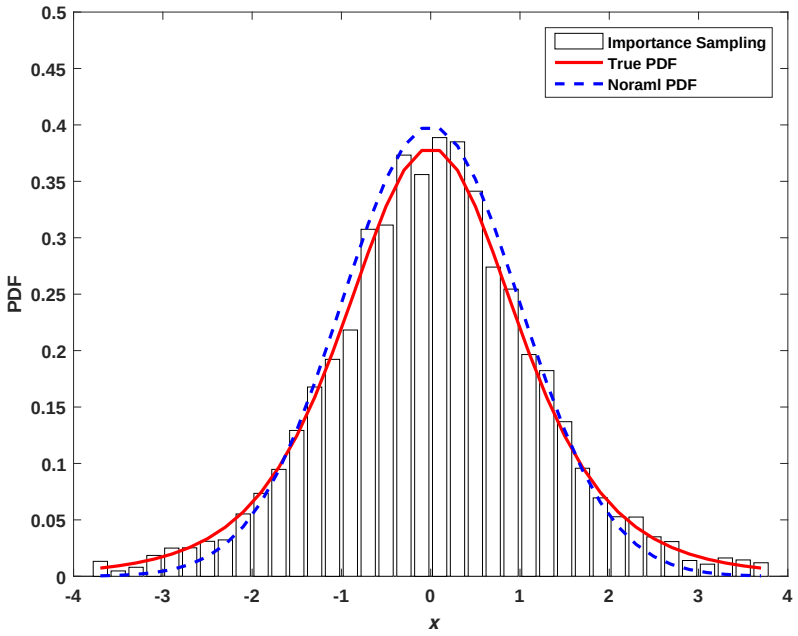


그림 3.4.19. 요점추출법과 t샘플 (MATLAB)

예제 3.4.22 R언어를 사용해서 예제 3.4.21의 문제를 풀기 위해서, 다음 R프로그램 다음 R프로그램 CrudeImportSample102R.R을 실행해 보자.

```
# ------------------------------------------------------------------------------
# Filename: CrudeImportSample102R.R
#  Generate t-random numbers using an Importance sampling
# Programmed by CBS
# ------------------------------------------------------------------------------
# Simulation
set.seed(11)
Ns <- 20000                    # Number of Samples
df <- 5
zz <- rnorm(Ns)
ratioo <- dt(zz, df, ncp=0, log=FALSE)/dnorm(zz)
tt <- ratioo*zz

# Plotting
# install.packages("ggplot2")
library(ggplot2)
setEPS()
plot.new()
postscript('CrudeImportSample102R.eps')  # Start to save figure
xx <- seq(-4,4,len=301)
tTrue <- dt(zz, df, ncp=0, log=FALSE)
normalTrue <- dnorm(zz)
RVdata1 <- data.frame(zz,tt,tTrue,normalTrue)
ggplot(RVdata1,aes(x=tt)) +
 geom_histogram(aes(y = ..density..),binwidth=0.2,fill="white",col="black") +
 xlim(-4,4) +
 geom_line(aes(x=zz,y=tTrue),lwd=1.2,col="red") +
 geom_line(aes(x=zz,y=normalTrue),lwd=1.2,col="blue",linetype=2)
dev.off()            # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 요점추출법을 적용해서 20000개 t샘플들을 생성한다. 이렇게 생성된 자유도가 5인 t샘플들의 히스토그램이 그림 3.4.20에 그려져 있다. 그림 3.4.20에서 백색 막대는 이 히스토그램이고 적색 실선은 자유도가 5인 t확률밀도함수이고, 청색 긴점선은 표준정규확률밀도함수이다. ■

제3.5절 MCMC입문

Markov체인몬테카를로법(Markov chain Monte Carlo method: MCMC method)은 보통 몬테카를로법에서는 하기 어려운 샘플의 생성과 기대값 계산을 Markov체인을 이용해서 시행하는 방법이다. 이 절에서는 MCMC법에 대해서 간단히 다루고자 한다. 좀 더 자세한 내용은 본서의 후속서인 *Bayesian Mehtods for Finance and Economics*에서 다룰 예정이다.

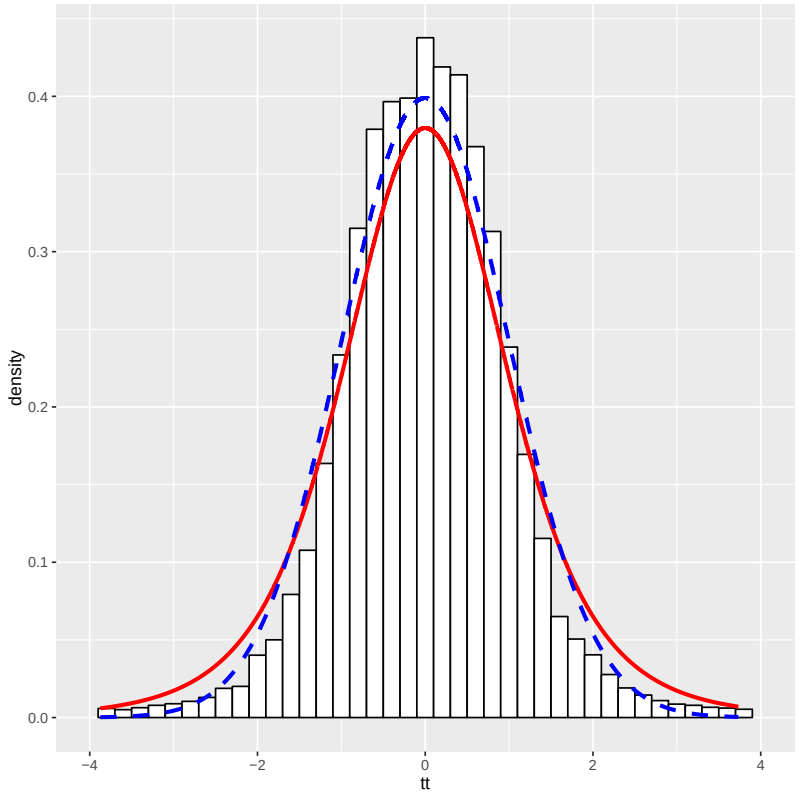


그림 3.4.20. 요점추출법과 t 샘플 (R)

3.5.1 Markov체인

시점 t에 따라 움직이는 확률변수열 $\{x_t\}$를 생각해 보자. 일반적으로, 시점 t에서 관측되는 확률변수 x_t의 확률밀도함수 $f(x_t)$는 시점 t 전에 관측되는 $x_{t-1}, x_{t-2}, \cdots$에 의존해서 결정된다. 그러나, 경우에 따라서는 x_t의 확률밀도함수가 x_{t-1}에만 의존하는 경우가 있다. 즉, 다음 식이 성립하는 경우가 있다.

$$f(x_t \mid x_{t-1}, x_{t-2}, \cdots) = f(x_t \mid x_{t-1}) \tag{3.5.1}$$

이러한 확률과정 $\{x_t\}$는 Markov성을 갖는다고 한다. 서로 독립인 확률변수열 $\{x_t\}$는 다음 식을 만족한다.

$$f(x_t \mid x_{t-1}, x_{t-2}, \cdots) = f(x_t) \tag{3.5.2}$$

따라서, Markov성은 독립성을 약간 느슨하게 만든 성질이라고 할 수 있다. 엄밀하게 말한다면, 식 (3.5.1)을 만족하는 확률과정 $\{x_t\}$는 1차 Markov성을 갖는다고 하고, 다음 식을 만족하면

확률과정 $\{x_t\}$는 k차 Markov성을 갖는다고 한다.

$$f(x_t \mid x_{t-1}, x_{t-2}, \cdots) = f(x_t \mid x_{t-1}, x_{t-2}, \cdots, x_{t-k}) \tag{3.5.3}$$

그러나, 식 (3.5.3)을 만족하는 $\{x_t\}$를 확률벡터과정 $\{\boldsymbol{x}_t \doteq [x_t, x_{t-2}, \cdots, x_{t-k}]\}$로 표현하면, 이 확률벡터과정은 1차 Markov성을 갖는다. 이후 식 (3.5.1)을 만족하는 확률과정, 즉 1차 Markov확률과정만을 다루기로 하자. 식 (3.5.1)의 조건부확률밀도함수가 시점 t에 의존하지 않으면, 확률과정 $\{x_t\}$가 정상적(stationary)이라 한다. 본서에서는 정상적 확률과정만을 다루기로 하자. 만약 확률변수 x_t의 지지대인 상태공간(state space) S가 이산집합이면, $\{x_t\}$를 이산형 Markov과정 또는 Markov체인(Markov chain)라 부른다. 이 절에서는 상태공간 S가 유한집합 $\{1, 2, \cdots, N\}$인 경우만을 다루기로 하자. Markov체인에 관한 자세한 내용은 최병선[3]의 제5.5절을 참조하라.

상태공간이 $S = \{1, 2, \cdots, N\}$인 정상적 Markov체인 $\{x_t\}$에서 상태 i에서 상태 j가 되는 추이확률(transition probability)을 다음과 같이 표기하자.

$$p(i, j) \doteq Pr(x_t = j \mid x_{t-1} = i), \;\; (i, j = 1, 2, \cdots, N) \tag{3.5.4}$$

이 Markov체인의 추이확률행렬(transition probability matrix)을 다음과 같이 정의하자.

$$\mathbb{P} \doteq \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \vdots \\ N \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & \cdots & N \end{array} \\ \begin{bmatrix} p(1,1) & p(1,2) & p(1,3) & \cdots & p(1,N) \\ p(2,1) & p(2,2) & p(2,3) & \cdots & p(2,N) \\ p(3,1) & p(3,2) & p(3,3) & \cdots & p(3,N) \\ \vdots & \vdots & \vdots & & \vdots \\ p(N,1) & p(N,2) & P(N,3) & \cdots & p(N,N) \end{bmatrix} \end{array} \tag{3.5.5}$$

이 $\mathbb{P}$를 간단히 추이행렬이라 부른다. 어떤 시점 t의 상태 i에서 다음 시점 $t+1$의 어떠한 상태로 추이되므로, 추이행렬 P에서 각 행의 원소들의 합은 1이다. 따라서, 다음 식이 성립한다.

$$\sum_{j=1}^{N} p(i, j) = 1, \;\; (i = 1, 2, \cdots, N) \tag{3.5.6}$$

즉, 식 $P\mathbf{1} = \mathbf{1}$이 성립한다. 여기서 $\mathbf{1} = [1, 1, \cdots, 1]^t$은 각 원소가 1인 N차원 열벡터이다. 식

$P\mathbf{1}=\mathbf{1}$에서 알 수 있듯이, 이 열벡터 $\mathbf{1}$은 P의 우고유벡터이고, 이에 대응하는 고유값은 1이다. 이 고유값 1은 P의 고유값들 중에서 절대값이 최대인 것이다.

시점 t의 상태 i에서 시점 $t+m$의 상태 j로 추이하는 확률을 제m단계 추이확률($m-$step transition probability)이라 하고, $p^{(m)}(i,j)$으로 표기하자. 또한, 제m단계 추이행렬을 다음과 같이 정의하자.

$$\mathbb{P}^{(m)} \doteq \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \vdots \\ N \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & \cdots & N \end{array} \\ \begin{bmatrix} p^{(m)}(1,1) & p^{(m)}(1,2) & p^{(m)}(1,3) & \cdots & p^{(m)}(1,N) \\ p^{(m)}(2,1) & p^{(m)}(2,2) & p^{(m)}(2,3) & \cdots & p^{(m)}(2,N) \\ p^{(m)}(3,1) & p^{(m)}(3,2) & p^{(m)}(3,3) & \cdots & p^{(m)}(3,N) \\ \vdots & \vdots & \vdots & & \vdots \\ p^{(m)}(N,1) & p^{(m)}(N,2) & p^{(m)}(N,3) & \cdots & p^{(m)}(N,N) \end{bmatrix} \end{array} \tag{3.5.7}$$

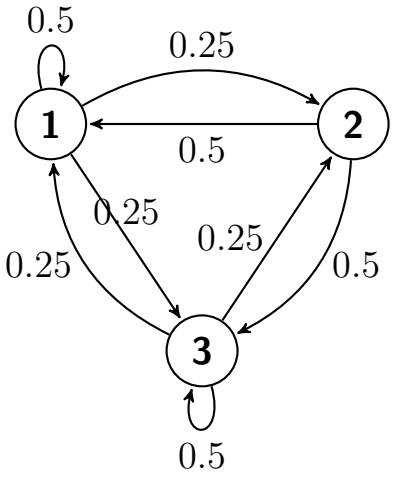
다음 식이 성립함을 쉽게 증명할 수 있다.

$$\mathbb{P}^{(m)} = \mathbb{P}^m \tag{3.5.8}$$

예제 3.5.1 아이스크림가게 Bing3에서는 바닐라맛, 딸기맛, 그리고 쵸콜릿맛 아이스크림들을 판다고 하자. 매일 오는 단골손님이 다음 날 선택하는 아이스크림의 종류는 당일 선택에만 의존하며, 그 추이행렬이 다음과 같다고 하자.

$$\mathbb{P} = \begin{array}{c} \\ V \\ S \\ C \end{array} \begin{array}{c} \begin{array}{ccc} V & S & C \end{array} \\ \begin{bmatrix} \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \end{bmatrix} \end{array} \tag{1}$$

예를 들어, 오늘 바닐라맛을 선택한 사람이 내일 딸기맛을 선택할 확률은 0.25이고, 오늘 딸기맛을 선택한 사람이 내일도 딸기맛을 선택할 확률은 0이며, 오늘 쵸콜릿맛을 선택한 사람이 내일 또 쵸콜릿맛을 선택할 확률은 0.5이다. 이 Markov체인의 상태추이도는 다음과 같다.



식 (3.5.8)을 적용하면, 각 $m(=2,3,4)$에 대해 이 Markov과정의 제m단계 추이행렬이 다음과 같음을 알 수 있다.

$$\mathbb{P}^{(2)} = \frac{1}{16}\begin{bmatrix} 7 & 3 & 6 \\ 6 & 4 & 6 \\ 6 & 3 & 7 \end{bmatrix},\ \mathbb{P}^{(3)} = \begin{bmatrix} 26 & 13 & 25 \\ 26 & 12 & 26 \\ 25 & 13 & 26 \end{bmatrix},\ \mathbb{P}^{(4)} = \frac{1}{256}\begin{bmatrix} 103 & 51 & 102 \\ 102 & 52 & 102 \\ 102 & 51 & 103 \end{bmatrix} \tag{2}$$

■

시점 t에서 확률변수 x_t가 상태 i를 취하는 확률을 다음과 같이 표기하자.

$$\pi(i)^{(t)} \doteq Pr(x_t = i) \tag{3.5.9}$$

상태집합 S에서 이 확률들을 모아서 다음 행벡터를 정의하자.

$$\boldsymbol{\pi}^{(t)} \doteq \left[\pi_1^{(t)}, \pi_2^{(t)}, \cdots, \pi_N^{(t)}\right] \tag{3.5.10}$$

특히, $t=0$에서 행벡터 $\boldsymbol{\pi}^{(0)} = \left[\pi_1^{(0)}, \pi_2^{(0)}, \cdots, \pi_N^{(0)}\right]$를 초기분포(initial distribution)라고 부른다. 이 행벡터는 확률질량함수에 해당하나 관습적으로 분포라 부른다. 각 t에서 다음 식이 성립함은 명백하다.

$$\boldsymbol{\pi}^{(t)}\mathbf{1} = 1 \tag{3.5.11}$$

시점 $t+1$에서 상태가 j일 확률은 다음과 같다.

$$\pi(j)^{(t+1)} = \sum_{i=1}^{N} Pr\left(x_{t+1} = j \mid x_t = i\right) Pr(x_t = i) = \sum_{i=1}^{N} \pi(i)^{(t)} p(i,j) \tag{3.5.12}$$

식 (3.5.12)를 다음과 같이 쓸 수 있다.

$$\boldsymbol{\pi}^{(t+1)} = \boldsymbol{\pi}^{(t)} \mathbb{P} \tag{3.5.13}$$

식 (3.5.8)과 식 (3.5.13)에서 알 수 있듯이, 다음 식들이 성립한다.

$$\boldsymbol{\pi}^{(t+m)} = \boldsymbol{\pi}^{(t)} \mathbb{P}^{(m)} = \boldsymbol{\pi}^{(t)} \mathbb{P}^{m} \tag{3.5.14}$$

식 (3.5.14)를 Chapman-Kolmogorov방정식이라 부른다.

상태추이를 무한 회 반복했을 때 극한에서 존재확률을 나타내는 벡터를 $\boldsymbol{\pi}$라고 하자. 즉, 다음 식이 성립한다고 하자.

$$\boldsymbol{\pi} \doteq \lim_{m \to \infty} \boldsymbol{\pi}^{(m)} \tag{3.5.15}$$

이 $\boldsymbol{\pi}$를 불변분포(invariance distribution) 또는 정상분포(stationary distribution)라 한다. 식 (3.5.14)에서 알 수 있듯이, 다음 식들이 성립한다.

$$\boldsymbol{\pi} = \lim_{m \to \infty} \boldsymbol{\pi}^{(0)} \mathbb{P}^{m} = \lim_{m \to \infty} \boldsymbol{\pi}^{(0)} \mathbb{P}^{m-1} \mathbb{P} = \boldsymbol{\pi}^{(0)} \mathbb{P}^{\infty} \tag{3.5.16}$$

식 (3.5.16)에서 알 수 있듯이, 불변분포 $\boldsymbol{\pi}$는 다음 식을 만족한다.

$$\boldsymbol{\pi} \mathbb{P} = \boldsymbol{\pi} \tag{3.5.17}$$

식 (3.5.17)에서 알 수 있듯이, 불변분포 $\boldsymbol{\pi}$는 추이행렬 $\mathbb{P}$의 고유값 1에 대응하는 좌고유벡터이다. 따라서, 불변분포는 초기분포에 의존하지 않음을 알 수 있다. 식 (3.5.11)에서 알 수 있듯이, 다음 식이 성립한다.

$$\boldsymbol{\pi} \mathbf{1} = 1 \tag{3.5.18}$$

식 (3.5.16)에서 알 수 있듯이, 극한행렬 $\mathbb{P}^{\infty}$는 각 행이 불변분포 $\boldsymbol{\pi}$인 행렬이다. 즉, 다음

식들이 성립한다.

$$\mathbb{P}^{\infty} = \begin{bmatrix} \boldsymbol{\pi} \\ \boldsymbol{\pi} \\ \vdots \\ \boldsymbol{\pi} \end{bmatrix} = \mathbf{1}\boldsymbol{\pi} \tag{3.5.19}$$

식 (3.5.19)의 우변의 $\mathbf{1}\boldsymbol{\pi}$는 랭크가 1인 행렬이다. 식 (3.5.17)에서 알 수 있듯이, 1은 행렬 $\mathbb{P}$의 고유값이다. 나머지 고유값들을 $\lambda_2, \lambda_3, \cdots, \lambda_N$이라 하자. 식 (3.5.19)에서 알 수 있듯이, 다음 식들이 성립해야 한다.

$$|\lambda(i)| < 1, \quad (i = 2, 3, \cdots, N) \tag{3.5.20}$$

만약 식 (3.5.20)이 성립하지 않으면, 극한행렬 $\mathbb{P}^{\infty}$는 수렴하지 않는다. 즉, 식 (3.5.19)가 성립하지 않는다. 행렬 $\mathbb{P}^m$의 고유값들은 $1^m, \lambda_2^m, \lambda_3^m, \cdots, \lambda_N^m$이다. 따라서, 행렬 $\mathbb{P}^m$이 행렬 $\mathbb{P}^{\infty}$에 접근하는 속도는 행렬 $\mathbb{P}$의 고유값들 중에서 절대값이 2번째로 큰 고유값에 의해 결정된다.

예제 3.5.2 예제 3.5.1의 Markov과정을 다시 살펴보자.

어떤 고객이 첫날 딸기맛 아이스크림을 샀다면, 이 고객의 초기분포는 다음과 같다.

$$\boldsymbol{\pi}^{(0)} = [0, 1, 0] \tag{1}$$

Chapman-Kolmogorov방정식 (3.5.14)을 적용하면, 이 고객이 이후 각 아이스크림을 선택할 확률벡터들은 다음과 같음을 알 수 있다.

$$\boldsymbol{\pi}(1) = \left[\frac{1}{2}, 0, \frac{1}{2}\right] \tag{2}$$

$$\boldsymbol{\pi}(2) = \left[\frac{3}{8}, \frac{1}{4}, \frac{3}{8}\right] \tag{3}$$

$$\boldsymbol{\pi}(3) = \left[\frac{13}{32}, \frac{3}{16}, \frac{13}{32}\right] \tag{4}$$

$$\boldsymbol{\pi}(9) = \left[\frac{52429}{131072}, \frac{13107}{65536}, \frac{52429}{131072}\right] \tag{5}$$

$$\boldsymbol{\pi}(10) = \left[\frac{2}{5}, \frac{52429}{262144}, \frac{2}{5}\right] \tag{6}$$

$$\boldsymbol{\pi}(11) = \left[\frac{2}{5}, \frac{1}{5}, \frac{2}{5}\right] \tag{7}$$

식 (7)은 컴퓨터의 유효숫자 안에서 반올림한 것이다. 식 (7)과 식 (3.5.13)에서 알 수 있듯이, 다음 식들이 성립한다.

$$\boldsymbol{\pi}^{(m+1)} = \boldsymbol{\pi}^{(m)}\mathbb{P} = \begin{bmatrix} \frac{2}{5}, & \frac{1}{5}, & \frac{2}{5} \end{bmatrix} \begin{bmatrix} \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \end{bmatrix} = \begin{bmatrix} \frac{2}{5}, & \frac{1}{5}, & \frac{2}{5} \end{bmatrix}, \quad (m \geq 11) \tag{8}$$

식 (8)에서 알 수 있듯이, 이 Markov과정의 불변분포는 다음과 같다.

$$\boldsymbol{\pi} = \begin{bmatrix} \frac{2}{5}, & \frac{1}{5}, & \frac{2}{5} \end{bmatrix} \tag{9}$$

즉, 아이스크림가게 Bing3에는 바닐라맛, 딸기맛, 그리고 쵸콜릿맛을 선택하는 확률들이 대략 $0.4, 0.2$그리고 0.4이다. 식 (3.5.8)을 적용하면, 다음 식이 성립함을 알 수 있다.

$$\mathbb{P}^{\infty} = \begin{bmatrix} \frac{2}{5} & \frac{1}{5} & \frac{2}{5} \\ \frac{2}{5} & \frac{1}{5} & \frac{2}{5} \\ \frac{2}{5} & \frac{1}{5} & \frac{2}{5} \end{bmatrix} \tag{10}$$

식 (9)와 식 (10)에서 식 (3.5.19)가 성립함을 확인할 수 있다.

식 (3.5.17)에서 알 수 있듯이, 불변분포 $\boldsymbol{\pi}$는 추이행렬 P의 고유값 1에 대응하는 좌고유벡터이다. 보편적인 상용프로그램에서는 우고유벡터만을 구할 수 있다. 따라서, 좌고유벡터를 구하기 위해서 식 (3.5.17)을 다음과 같이 전치를 하자.

$$\mathbb{P}^t \boldsymbol{\pi}^t = \boldsymbol{\pi}^t \tag{11}$$

추이행렬 P의 고유값들과 전치행렬 P^t의 고유값들은 같다. 따라서, 행렬 P^t의 우고유벡터를 구한 다음 전치를 시키면, 추이행렬 P의 좌고유벡터이다. MATLAB함수 eig.m을 사용해서 구한 행렬 P^t의 고유값들과 우고유벡터들 $\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3$는 다음과 같다.

$$\lambda_1 = 1, \quad \lambda_2 = \frac{1}{4}, \quad \lambda_3 = -\frac{1}{4} \tag{12}$$

$$\boldsymbol{y}_1 = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix}, \quad \boldsymbol{y}_2 = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}, \quad \boldsymbol{y}_3 = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} \tag{13}$$

식 (12)에서 알 수 있듯이, 다음 식들이 성립한다.

$$|\lambda_2| < 1, \quad |\lambda_3| < 1 \tag{14}$$

불변분포 $\boldsymbol{\pi}$는 추이행렬 P의 고유값 1에 대응하는 좌고유벡터 $\boldsymbol{y}_1$에 해당한다. 식 (13)에서 알 수 있듯이, 다음 식이 성립한다.

$$\boldsymbol{\pi} = \frac{1}{5}\boldsymbol{y}_1 = \begin{bmatrix} \frac{2}{5} & \frac{1}{5} & \frac{2}{5} \end{bmatrix} \tag{15}$$

식 (15)는 식 (9)와 같은 결과를 보여준다. ■

3.5.2 Gibbs샘플링

확률벡터 $\boldsymbol{x} = [x_1, x_2, \cdots, x_p]^t$의 결합확률밀도함수를 $f(\boldsymbol{x}) = f(x_1, x_2, \cdots, x_p)$라 하자. 또한, 확률변수 $x(i)$를 제외한 다른 확률변수들로 이루어진 확률벡터를 $\boldsymbol{x}_{-i} \doteq [x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_p]^t$로 표기하자. 확률벡터 $\boldsymbol{x}_{-i}$가 주어진 조건 하에서 확률변수 $x(i)$의 조건부확률밀도함수를 $f(x(i) \mid \boldsymbol{x}_{-i})$로 표기하자. 확률밀도함수 $f(\boldsymbol{x})$에서 샘플을 추출하는 것은 어렵지만, 조건부확률밀도함수 $f(x(i) \mid \boldsymbol{x}_{-i})$에서 샘플의 추출이 쉬운 경우에는 Gibbs샘플링을 사용해서 확률벡터 $\boldsymbol{x}$의 결합확률밀도함수로부터 샘플벡터들을 생성한다. 이 절에서는 Gibbs샘플링을 간단히 소개하고자 한다.

Gibbs샘플링은 다음 알고리즘을 적용하는 것이다.

알고리즘 3.5.1: Gibbs샘플링

(1단계) 초기벡터 $\boldsymbol{x}^{(0)} = \left[x_1^{(0)}, x_2^{(0)}, \cdots, x_p^{(0)}\right]^t$를 선택한다.

(2단계) 제j번째 관찰벡터 $\boldsymbol{x}^{(j)} = \left[x_1^{(j)}, x_2^{(j)}, \cdots, x_p^{(j)}\right]^t$가 주어졌을 때, 다음 식들을

사용해서 제$j+1$번째 관찰벡터 $\boldsymbol{x}^{(j+1)} = \left[x_1^{(j+1)}, x_2^{(j+1)}, \cdots, x_p^{(j+1)}\right]^t$를 생성한다.

$$
\begin{aligned}
x_1^{(j+1)} &\overset{d}{\sim} f\left(x_1 \,\middle|\, x_2^{(j)}, x_3^{(j)}, \cdots, x_{p-1}^{(j)}, x_p^{(j)}\right) \\
x_2^{(j+1)} &\overset{d}{\sim} f\left(x_2 \,\middle|\, x_1^{(j+1)}, x_3^{(j)}, \cdots, x_{p-1}^{(j)}, x_p^{(j)}\right) \\
&\vdots \\
x_{p-1}^{(j+1)} &\overset{d}{\sim} f\left(x_p \,\middle|\, x_1^{(j+1)}, x_2^{(j+1)}, \cdots, x_{p-2}^{(j+1)}, x_p^{(j)}\right) \\
x_p^{(j+1)} &\overset{d}{\sim} f\left(x_p \,\middle|\, x_1^{(j+1)}, x_2^{(j+1)}, \cdots, x_{p-2}^{(j+1)}, x_{p-1}^{(j+1)}\right)
\end{aligned}
$$

(3단계) 만약 j가 원하는 표본수 n보다 작으면 제2단계로 다시 돌아가고, 그렇지 않으면, 알고리즘의 수행을 끝낸다.

알고리즘 3.5.1에 의해서, Markov체인의 실현벡터들 $\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)}, \cdots$ 가 얻어진다. 실제 알고리즘 3.5.1을 적용하기 위해서는 먼저 적절한 초기벡터 $\boldsymbol{x}^{(0)}$를 정한 다음, 원하는 표본개수 n보다 k개 많은 $n+k$개 샘플벡터들을 생성하여, 첫 k개 샘플벡터들은 버리고, 나머지 n개 샘플벡터들을 사용한다. 이러한 첫 k개를 번인기간(burn-in period)이라 부른다. 확률변수들 사이에 상관관계가 큰 경우에는 k를 크게 취할 필요가 있다.

예제 3.5.3 확률벡터 $[x, y]^t$가 다음 2변량 정규확률분포를 따른다고 하자.

$$
\begin{bmatrix} x \\ y \end{bmatrix} \overset{d}{\sim} \mathcal{N}\left(\begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} \right) \tag{1}
$$

다음 식들이 성립한다.

$$
y \mid x \overset{d}{\sim} \mathcal{N}\left(\mu_y + \rho\left[x - \mu_x\right], 1 - \rho^2\right) \tag{2}
$$

$$
x \mid y \overset{d}{\sim} \mathcal{N}\left(\mu_x + \rho\left[y - \mu_y\right], 1 - \rho^2\right) \tag{3}
$$

다음 식들을 반복적용해서 Gibbs샘플들 $\left\{\left[x^{(j)}, y^{(j)}\right]\right\}$을 생성한다.

$$
y^{(j+1)} \mid x^{(j)} \overset{d}{\sim} \mathcal{N}\left(\mu_y + \rho\left[x^{(j)} - \mu_x\right],\ 1 - \rho^2\right) \tag{4}
$$

$$
x^{(j+1)} | y^{(j+1)} \overset{d}{\sim} \mathcal{N}\left(\mu_x + \rho\left[y^{(j+1)} - \mu_y\right], 1 - \rho^2\right) \tag{5}
$$

식 (4)와 식 (5)를 사용해서 Gibbs샘플을 발생시키기 위해서, 다음 MATLAB프로그램 GibbsBivariateNormal101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
% Filename: GibbsBivariateNormal101.m
% Markov Chain Monte Carlo 1 for Bivariate Normal Distribution
% Programmed by CBS
% ------------------------------------------------------------------------
clear, clf, close all
nn = 1050;
x = zeros(nn,1); y = zeros(nn,1);
rho = 0.9;
mux = 1; muy = -1;
sigma = sqrt(1-rho^2);
rng(5489,'twister');   % Use twister or state for seed.
% Gibbs
x(1) = 10; y(1)=10;
for jj=2:nn
    muxdum = mux + rho*(y(jj-1)-muy);
    x(jj) = muxdum + sigma*randn(1);
    muydum = muy + rho*(x(jj)-mux);
    y(jj) = muydum + sigma*randn(1);
end
% Ploting
subplot(2,2,1)
plot(x(1:50),y(1:50),'r.','linewidth',1.5)
hold on
set(gca,'fontsize',11,'fontweigh','bold')
axis([-3 12 -4 11])
plot([-3 12],[-1 -1],'k-',[1 1],[-4 11],'k-')
title('\bf Initial Gibbs Samples')
hold off
subplot(2,2,2)
plot(x(501:550),y(501:550),'go','linewidth',1.5)
hold on
set(gca,'fontsize',11,'fontweigh','bold')
axis([-3 12 -4 11])
plot([-3 12],[-1 -1],'k-',[1 1],[-4 11],'k-')
title('\bf Middle Gibbs Samples')
hold off
subplot(2,2,3)
plot(x(1001:1050),y(1001:1050),'bd','linewidth',1.5)
hold on
set(gca,'fontsize',11,'fontweigh','bold')
axis([-3 12 -4 11])
plot([-3 12],[-1 -1],'k-',[1 1],[-4 11],'k-')
title('\bf Final Gibbs Samples')
hold off
% Generating Bivariate Normal Random Numbers using SVD
zmat = zeros(2,nn);
mu = [ mux  muy ]';
Sigma = [ 1 rho ; rho 1 ];
[ P,D,Q ] = svd(Sigma)       % Schur decomposition
srLamb = sqrt(D)
for j=1:nn
    z = mu + P*srLamb*randn(2,1);
    zmat(:,j) = z;
end
% Ploting
subplot(2,2,4)
```

```
plot(zmat(1,1001:1050),zmat(2,1001:1050),'bd',...
     zmat(1,501:550),zmat(2,501:550),'go', ...
     zmat(1,1:50),zmat(2,1:50),'r.','linewidth',1.5)
hold on
set(gca,'fontsize',11,'fontweigh','bold')
axis([-3 12 -4 11])
plot([-3 12],[-1 -1],'k-',[1 1],[-4 11],'k-')
title('\bf Samples via SVD')
hold off
saveas(gcf,'GibbsBivariateNormal101','epsc')
save('GibbsBivariateNormal101','x','y','zmat')
% End of program
% --------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 평균벡터가 $[\mu_x, \mu_y]^t = [1, -1]^t$이고 또한 상관계수가 $\rho = 0.9$인 2변량 정규확률분포에서 Gibbs샘플들 $\left\{ \left[x^{(j)}, y^{(j)}\right] \mid j = 1, 2, \cdots, 1050 \right\}$이 발생된다. 여기서 초기벡터는 $[10, 10]$이다. 이 Gibbs샘플들의 일부가 그림 3.5.1에 그려져 있다. 그림 3.5.1의 좌측상단 그래프는 첫 50개 Gibbs샘플들의 산점도이고, 우측상단 그래프는 중간 50개 Gibbs샘플들의 산점도이고, 좌측하단 그래프는 마지막 50개 Gibbs샘플들의 산점도이다. 이 그래프들에서 확인할 수 있듯이, 반복회수가 커짐에 따라 Gibbs샘플들은 점점 불변분포에 가까운 확률분포에서 생성되었다고 할 수 있다.

Cholesky분해, Schur분해, 그리고 특이값분해(Singular Value Decomposition: SVD)를 사용해서, 다변량정규확률분포로부터 정규난벡터(normal random boldor)를 생성할 수 있다. 이 중에서 특이값분해를 사용해서 식 (1)의 2변량 정규확률분포로부터 생성한 난벡터들의 일부가 그림 3.5.1의 우측하단 그래프에 그려져 있다. 이 그림에서 적색 점들은 첫 50개 샘플들이고, 녹색 원은 중간 50개 샘플들이고, 청색 다이아몬드는 마지막 50개 샘플들이다. 이 그래프에서 알 수 있듯이, 반복회수에 상관없이 특이값분해를 사용해서 발생시킨 정규난벡터들은 식 (1)의 2변량 정규확률분포를 따른다고 할 수 있다. ■

예제 3.5.4 R언어를 사용해서 예제 3.5.3의 문제를 풀기 위해서, 다음 R프로그램 다음 R프로그램 GibbsBivariateNormal101RR.R을 실행해 보자.

```
# --------------------------------------------------------------------------
# Filename: GibbsBivariateNormal101R.R
# Gibbs Sampling for Bivariate Normal Distribution
# Programmed by CBS
# --------------------------------------------------------------------------
# Set-up
set.seed(11)
nn <- 1050;
x <- y <- rep(0,nn)
rho <- 0.9;
```

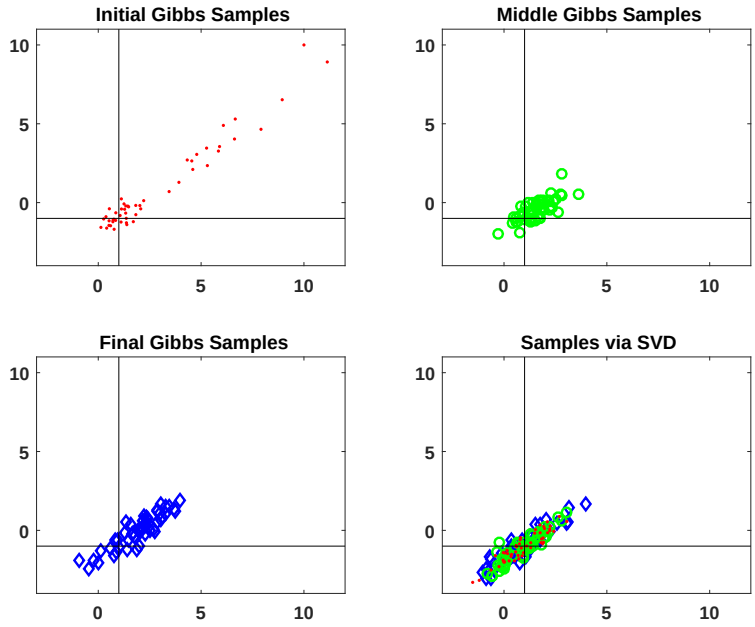


그림 3.5.1. Gibbs샘플러와 2변량 정규확률분포 (MATLAB)

```
mux <- 1; muy <- -1
sigma <- sqrt(1-rho^2)
# Gibbs Sampling
x[1] <- 10; y[1] <- 10;
for (jj in 2:nn){
    muxdum <- mux + rho*(y[jj-1]-muy)
    x[jj] <- muxdum + sigma*rnorm(1)
    muydum <- muy + rho*(x[jj]-mux)
    y[jj] <- muydum + sigma*rnorm(1)
}
# Generating Bivariate Normal Random Numbers using SVD
zmat <- rep(0,2*nn)
dim(zmat) <- c(2,nn)
( mu <- rbind(mux,muy) )
( Sigma <- matrix( c(1, rho, rho, 1), nrow=2 ) )

(SV <- svd(Sigma))
D <- diag(SV$d)
P <- SV$u
Q <- SV$v
( dum101 <- Sigma - P%*%D%*%t(Q) )
srLamb <- sqrt(D)
for (jj in 1:nn){
    z <- mu + P%*%srLamb%*%rnorm(2);
    zmat[,jj] <- z
}

# Plotting
# install.packages("ggplot2")
library(ggplot2)
# install.packages("grid")
library(grid)
setEPS()
plot.new()
postscript('GibbsBivariateNormal101R.eps')  # Start to save figure
x50 <- x[1:50]; y50 <- y[1:50]
```

```
RVdata50 <- data.frame(x50,y50)
plot11 <- ggplot(RVdata50,aes(x=x50,y=y50)) +
          geom_point(col="red",lwd=1.5,shape=16) +
          xlim(-1,12) + ylim(-4,11) +
          geom_hline(yintercept=-1,col="black",lwd=1) +
          geom_vline(xintercept=1,col="black",lwd=1)
x550 <- x[501:550]; y550 <- y[501:550]
RVdata550 <- data.frame(x550,y550)
plot12 <- ggplot(RVdata550,aes(x=x550,y=y550)) +
          geom_point(col="green",lwd=1.5,shape=16) +
          xlim(-1,12) + ylim(-4,11) +
          geom_hline(yintercept=-1,col="black",lwd=1) +
          geom_vline(xintercept=1,col="black",lwd=1)
x1050 <- x[1001:1050]; y1050 <- y[1001:1050]
RVdata1050 <- data.frame(x1050,y1050)
plot21 <- ggplot(RVdata1050,aes(x=x1050,y=y1050)) +
          geom_point(col="blue",lwd=1.5,shape=5) +
          xlim(-1,12) + ylim(-4,11) +
          geom_hline(yintercept=-1,col="black",lwd=1) +
          geom_vline(xintercept=1,col="black",lwd=1)
z50 <- zmat[,c(1:50)]
z550 <- zmat[,c(501:550)]
z1050 <- zmat[,c(1001:1050)]
zz <- cbind(z50,z550,z1050)
zx <- zz[1,]; zy <- zz[2,]
RVdataSVD <- data.frame(zx,zy)
plot22 <- ggplot(RVdataSVD,aes(x=zx,y=zy)) +
          geom_point(col="dark blue",lwd=1.5,shape=1) +
          xlim(-1,12) + ylim(-4,11) +
          geom_hline(yintercept=-1,col="black",lwd=1) +
          geom_vline(xintercept=1,col="black",lwd=1)
pushViewport(viewport(layout=grid.layout(2,2)))
print(plot11, vp=viewport(layout.pos.row=1, layout.pos.col=1))
print(plot12, vp=viewport(layout.pos.row=1, layout.pos.col=2))
print(plot21, vp=viewport(layout.pos.row=2, layout.pos.col=1))
print(plot22, vp=viewport(layout.pos.row=2, layout.pos.col=2))
dev.off()            # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 평균벡터가 $[\mu_x, \mu_y]^t = [1, -1]^t$ 이고 또한 상관계수가 $\rho = 0.9$ 인 2변량 정규확률분포에서 Gibbs샘플들 $\left\{ \left[x^{(j)}, y^{(j)}\right] \mid j = 1, 2, \cdots, 1050 \right\}$ 이 발생된다. 여기서 초기벡터는 $[10, 10]$ 이다. 이 Gibbs샘플들의 일부가 그림 3.5.2에 그려져 있다. 그림 3.5.2의 좌측상단 그래프는 첫 50개 Gibbs샘플들의 산점도이고, 우측상단 그래프는 중간 50개 Gibbs샘플들의 산점도이고, 좌측하단 그래프는 마지막 50개 Gibbs샘플들의 산점도이다. 이 그래프들에서 확인할 수 있듯이, 반복회수가 커짐에 따라 Gibbs샘플들은 점점 불변분포에 가까운 확률분포에서 생성되었다고 할 수 있다. 또한, 특이값분해를 사용해서 주어진 2변량 정규확률분포로부터 생성한 난벡터들의 일부가 그림 3.5.2의 우측하단 그래프에 그려져 있다. 이 그림에서 첫 50개 샘플들, 중간 50개 샘플들 그리고 마지막 50개 샘플들이 짙은 청색 원으로 표시되어 있다. 이 그래프에서 알 수 있듯이, 반복회수에 상관없이 특이값분해를 사용해서 발생시킨 정규난벡터들은 2변량 정규확률분포를 따른다고 할 수 있다. ■

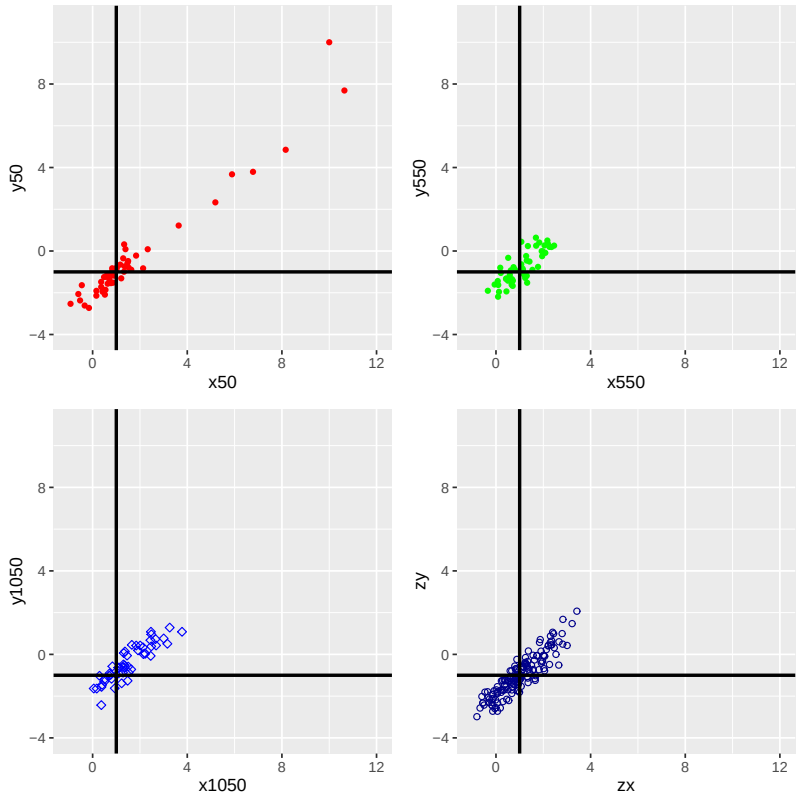


그림 3.5.2. Gibbs샘플러와 2변량 정규확률분포 (R)

3.5.3 Metropolis-Hastings샘플링

음이 아닌 상수들 $c(1), c(2), \cdots, c(N)$에서 다음 식들을 만족하는 확률밀도함수 $\{\pi(1), \pi(2), \cdots, \pi(N)\}$을 생성할 수 있다.

$$\pi(i) = \frac{c(i)}{C}, \quad (i = 1, 2, \cdots, N) \tag{3.5.21}$$

여기서 기준화상수 $C \doteq \sum_{i=1}^{N} c(i)$를 쉽게 구할 수 없다고 하자. 예를 들어, 베이즈통계학에서 사후확률분포의 기준화상수를 구하는 것이 쉽지 않다. 우리 문제는 주어진 $c(1), c(2), \cdots, c(N)$을 바탕으로 확률밀도함수 $\{\pi(1), \pi(2), \cdots, \pi(N)\}$으로부터 샘플들을 생성하는 것이다.

먼저 Markov체인의 시간가역성(time reversibility)에 대해서 살펴보자. 각 $i \neq j$에 대해 다음 식이 성립하면, Markov체인는 시간적으로 가역이라 한다.

$$\pi(i)p(i,j) = \pi(j)p(j,i) \tag{3.5.22}$$

Ross [42]의 제4.7절에서 알 수 있듯이, 시간적으로 가역인 Markov체인은 다음 식을 만족한다.

$$\boldsymbol{\pi} = \boldsymbol{\pi}\mathbb{P} \tag{3.5.23}$$

즉, 시간가역적 Markov체인에는 불변분포가 존재한다.

상태공간이 $S = \{1, 2, \cdots, N\}$이며 축소불가능하고(irreducible) 양재귀적이며(positive recurrent) 비주기적인(aperiodic)인 Markov체인의 추이행렬 $\mathbb{Q} = [q(i,j)]$로부터 불변분포가 $\{\pi(1), \pi(2), \cdots, \pi(N)\}$인 Markov체인 $\{x_t\}$를 다음과 같이 생성할 수 있다.

알고리즘 3.5.2: Metropolis-Hastings샘플러

(1단계) 초기벡터 x_0를 선택한다.

(2단계) 만약 시점 t에서 상태가 i이면, 즉 $x_t = i$이면, 다음 식을 만족하는 확률변수 y의 실현값 j를 생성한다.

$$Pr(y = j) = q(i,j), \quad (j = 1, 2, \cdots, N)$$

(3단계) 확률 $\alpha(i,j)$로 $x_{t+1} = j$라 하고, 확률 $1 - \alpha(i,j)$로 $x_{t+1} = i$라 하자. 여기서 $\alpha(i,j)$는 다음과 같다.

$$\alpha(i,j) \doteq \min\left\{\frac{c(j)q(j,i)}{c(i)q(i,j)}, 1\right\}$$

(4단계) 만약 t가 원하는 표본수 n보다 작으면, t를 $t+1$으로 바꾼 다음 제2단계로 다시 돌아간다. 그렇지 않으면, 알고리즘의 수행을 끝낸다.

알고리즘 3.5.2를 Metropolis-Hastings샘플러 또는 MH샘플러라 부른다. MH샘플러에서 발생하는 $\{x_t\}$는 추이확률이 다음과 같은 Markov체인이다.

$$p(i,j) = q(i,j)\alpha(i,j), \quad (j \neq i) \tag{3.5.24}$$

$$p_{ii} = q(i,i) + \sum_{k \neq i} q(i,k)\left[1 - \alpha(i,k)\right] \tag{3.5.25}$$

이 Markov체인이 시간적으로 가역이면, 식 (3.5.24)에서 알 수 있듯이 식 (3.5.22)와 다음 식은 동치이다.

$$\pi(i)q(i,j)\alpha(i,j) = \pi(j)q(j,i)\alpha(j,i) \tag{3.5.26}$$

만약 다음 식이 성립하면, 식 (3.5.26)이 성립한다.

$$\alpha(i,j) = \min\left\{\frac{c(j)q(j,i)}{c(i)q(i,j)}, 1\right\} \tag{3.5.27}$$

즉, 상태 i에서 상태 j로 추이할 확률과 반대로 상태 j에서 상태 i로 추이할 확률의 비의 확률로 추이의 유무가 결정지어진다. 시간가역적인 Markov체인 $\{x_t\}$는 $\{\pi(1), \pi(2), \cdots, \pi(N)\}$을 불변분포로 갖는다. 여기서 중요한 점은 알고리즘 3.5.2에는 기준화상수 C가 나타나지 않는다는 것이다. 알고리즘 3.5.1의 Gibbs샘플링은 알고리즘 3.5.2의 Metropolis-Hastings샘플러에 있어서 항상 추이가 되는 특수한 경우이다. 만약 추이행렬 $\mathbb{Q} = \{q(i,j)\}$가 대칭행렬이면, 식 (3.5.26)을 다음과 같이 쓸 수 있다.

$$\alpha(i,j) = \min\left(\frac{c(j)}{c(i)}, 1\right) \tag{3.5.28}$$

식 (3.5.28)을 사용하는 Metropolis-Hastings샘플러를 Metropolis샘플러라 부른다.

예제 3.5.5 다음과 같은 확률밀도함수를 살펴보자.

$$\pi(\theta) \propto \theta^{37}[1-\theta]^{63}, \quad \left(\theta = \frac{1}{15}, \frac{2}{15}, \frac{3}{15}, \frac{4}{15}, \frac{5}{15}\right) \tag{1}$$

이 확률밀도함수를 구하기 위해서는 다음 값을 계산해야 한다.

$$C = \sum_{k=1}^{5}\left[\frac{k}{15}\right]^{37}\left[\frac{15-k}{15}\right]^{63} \tag{2}$$

컴퓨터를 사용해서 다음 식이 성립함을 알 수 있다.

$$C = 1.979005048125769 \cdot 10^{-29} \tag{3}$$

즉, 확률밀도함수는 다음과 같다.

$$\pi(\theta) = \frac{1}{C}\theta^{37}[1-\theta]^{63}, \quad \left(\theta = \frac{1}{15}, \frac{2}{15}, \frac{3}{15}, \frac{4}{15}, \frac{5}{15}\right) \tag{4}$$

지금부터는 상수 C를 모른다고 가정하고 MH알고리즘을 사용해서 이 확률밀도함수로부터

샘플들을 생성하자. 다음 제안행렬을 생각해보자.

$$\mathbb{Q}_1 = \begin{bmatrix} 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \end{bmatrix} \tag{5}$$

제안행렬 $\mathbb{Q}_1$에서는 각 추이확률이 $q(i,j) = 0.2$이다. 따라서, 행렬 $\mathbb{Q}_1$이 Markov행렬임이 자하다. 또한, 다음 제안행렬을 생각해보자.

$$\mathbb{Q}_2 = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0 & 0.5 & 0 \\ 0 & 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 0.5 & 0.5 \end{bmatrix} \tag{6}$$

제안행렬 $\mathbb{Q}_2$에서는 상태가 0.1또는 0.9인 경우를 제외하고 반드시 이웃하는 상태로만 이동한다. 또한, 상태가 0.1또는 0.9인 경우에는 자신과 이웃하는 상태로만 이동한다. 따라서, 행렬 $\mathbb{Q}_2$도 Markov행렬이다. 행렬 $\mathbb{Q}_1$이나 행렬 $\mathbb{Q}_2$ 모두 대칭이므로, Metropolis샘플러를 이용할 수 있다.

Metropolis샘플러를 적용해서 이 확률밀도함수를 구하기 위해서, 다음 MATLAB프로그램 MCMCDiscreteUniform101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
% Filename: MCMCDiscreteUniform101.m
% Posterior Distribution of Discrete Uniform RV by MCMC
% Programmed by CBS
% ------------------------------------------------------------------------
function MCMCDiscreteUniform101
clear all, close all
% Proposed Matrices
Q1 = 0.2*ones(5,5);
Q2 = zeros(5,5);
Q2(1,1) = 0.5; Q2(5,5) = 0.5;
for kk=1:4
    Q2(kk,kk+1) = 0.5;
    Q2(kk+1,kk) = 0.5;
end
Q2
% MCMC
rand('twister',5489);
```

```
theta = (1:1:5)/15;
nSim =50;
BurnIn = 1950;
c = 1:5;
% Initial Value
x = GenerateDiscreteRS(theta,c/15,2)';
tt = 1;
% Iteration
for tt=2:(nSim+BurnIn)
    ii = round((x(:,tt-1)-0.1)/0.2+1);
    y(1) = GenerateDiscreteRS(theta,Q1(ii(1),:),1);
    y(2) = GenerateDiscreteRS(theta,Q2(ii(2),:),1);
    jj = round((y-0.1)/0.2+1);
    alpha = min(1,c(jj)./c(ii));
    u = rand(2,1);
    for kk=1:2
        if u(kk) <= alpha(kk)
           x(kk,tt) = y(kk);
        else
           x(kk,tt) = x(kk,tt-1);
        end
    end
end
timee = 1:50
theta1 = x(1,timee)
% Time Series Plot
subplot(2,1,1)
plot(timee,theta1,'g-',timee,theta1,'k*','linewidth',1.5)
set(gca,'fontsize',11,'fontweigh','bold')
axis([ 0 51 0 101/300 ])
xlabel('\bf Time','fontsize',12)
ylabel('\bf State by Q_{1}','fontsize',12)
theta2 = x(2,timee)
% Time Series Plot
subplot(2,1,2)
plot(timee,theta2,'b-',timee,theta2,'k*','linewidth',1.5)
set(gca,'fontsize',11,'fontweigh','bold')
axis([ 0 51 0 101/300 ])
xlabel('\bf Time','fontsize',12)
ylabel('\bf State by Q_{2}','fontsize',12)
saveas(gcf,'MCMCDiscreteUniform101b','epsc')
% Histogram
xHistL = min(theta);
xHistR = max(theta);
HistIncrement = 1/15;
PMFind = (xHistL:HistIncrement:xHistR )';
[Nbar1, xcenter1] = hist(x(1,BurnIn+1:BurnIn+nSim), PMFind);
[Nbar2, xcenter2] = hist(x(2,BurnIn+1:BurnIn+nSim), PMFind);
Nheight1 = Nbar1/nSim;
Nheight2 = Nbar2/nSim;
figure
subplot(1,2,1)
hold on
bar(xcenter1, Nheight1, 0.2, 'g');
set(gca,'fontsize',11,'fontweigh','bold','xlim',[1/30,11/30])
ylabel('\bf PMF by Q_{1}','fontsize',12)
hold off;
subplot(1,2,2)
hold on
bar(xcenter2, Nheight2, 0.2, 'b');
set(gca,'fontsize',11,'fontweigh','bold','xlim',[1/30,11/30])
```

```
ylabel('\bf PMF by Q_{2}','fontsize',12)
saveas(gcf,'MCMCDiscreteUniform101a','epsc')
save('MCMCDiscreteUniform101','x')
hold off;
end
% End of Program
% --------------------------------------------------------------------------
% Filename: GenerateDiscreteRS.m
% Generate Discrete Random Samples
% ---------------------------------------------------------------------
function RS = GenerateDiscreteRS(x,p,nrs)
% x = Support set (i.e., domain) of the distribution
% p = Probability density function
% nrs = Number of the generated random numbers
% (Ex) RS = GenerateDiscreteRS([1 3 5],[0.2 0.5 0.3],1200)
cumdf = cumsum(p);
RS = zeros(1,nrs);
for ii = 1:1:nrs
    u = rand;
    idum = 1;
    while cumdf(idum) <= u
          idum = idum+1;
    end;
    RS(ii) = x(idum);
end
end
% End of the function
% ---------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 각 제안행렬을 사용한 Metropolis샘플러를 적용해서 샘플들을 발생시킨다. 이때 번인기간은 1950이고 샘플들의 개수는 50이다. 각 제안행렬에 대한 시계열산점도가 그림 3.5.3에 그려져 있다. 그림 3.5.3에서 상단 그래프는 제안행렬 $\mathbb{Q}_1$에 의해 발생한 시간구간 $[1, 50]$에서 샘플들의 시계열산점도이고, 그리고 하단 그래프는 제안행렬 $\mathbb{Q}_2$에 의해 발생한 시간구간 $[1, 50]$에서 샘플들의 시계열산점도이다. 또한, 이렇게 발생된 확률밀도함수들이 그림 3.5.4에 그려져 있다. 그림 3.5.4에서 좌측 그래프는 제안행렬 $\mathbb{Q}_1$에 의해 발생한 확률밀도함수이고, 우측 그래프는 제안행렬 $\mathbb{Q}_2$에 의해 발생한 확률밀도함수이다. 이 그래프들에서 알 수 있듯이, 제안행렬 $\mathbb{Q}_2$보다 제안행렬 $\mathbb{Q}_1$이 더 좋은 MH샘플들을 생성한다. ■

예제 3.5.6 제2.3절에서 소개했듯이, 정규확률밀도함수들의 선형결합으로 이루어진 확률밀도함수가 혼합정규확률밀도함수이다. 이 예제에서는 다음과 같은 혼합정규확률밀도함수에서 샘플들을 생성하기로 하자.

$$f(x) = 0.6\, n\left(x \mid 1, 2^2\right) + 0.4\, n\left(x \mid 10, 3^2\right) \tag{1}$$

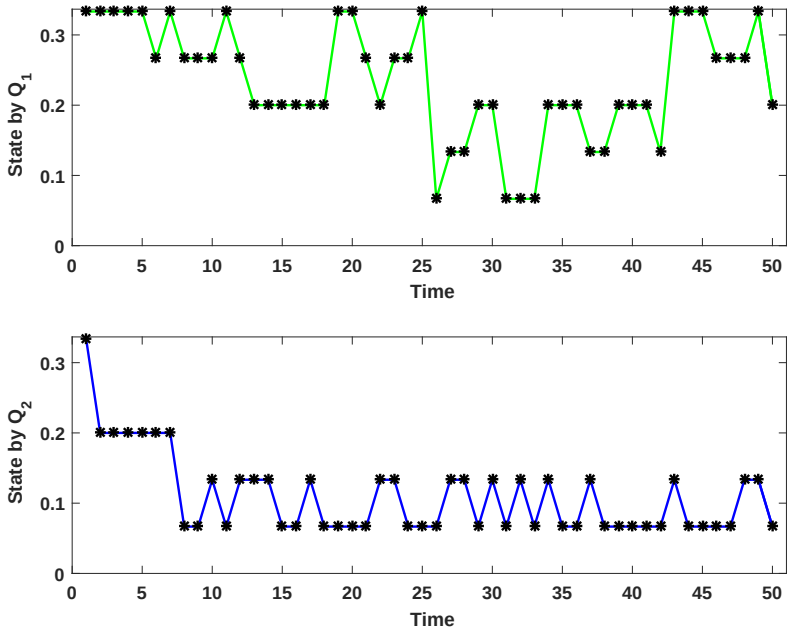


그림 3.5.3. MH샘플러에 의해 생성된 이산일양샘플들의 시계열산포도

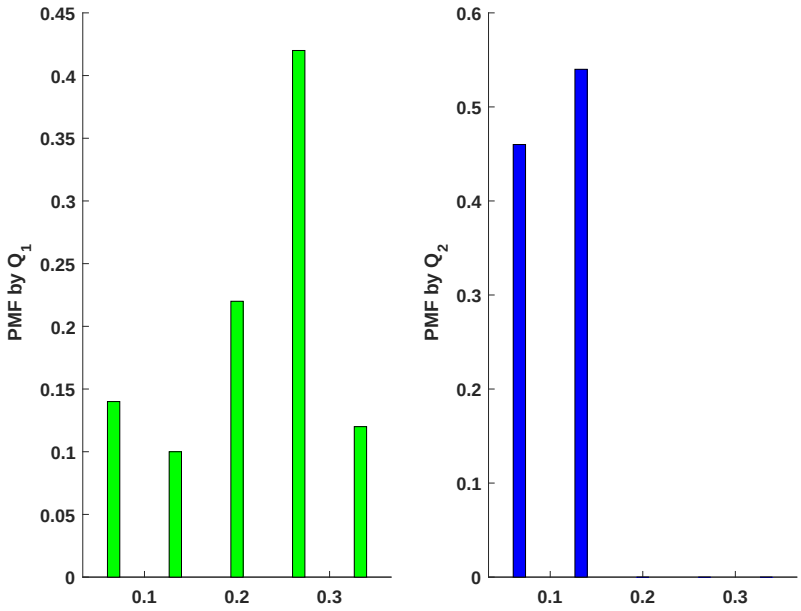


그림 3.5.4. MH샘플러에 의해 생성된 이산일양샘플들의 히스토그램

MH샘플러를 적용해서 이 확률밀도함수를 구하기 위해서, 다음 R프로그램 MHsampler-GaussianMixture101R.R을 실행해 보자.

```
# ------------------------------------------------------------------------
# Filename: MHsamplerGaussianMixture101R.R
# Metropolis-Hastings sampler for Gaussian Mixture
# Programmed by CBS
# ------------------------------------------------------------------------
# TargetDensity
mu1 <- 1; sig1 <- 2;
```

```
mu2 <- 10; sig2 <- 3;
w <- 0.6
Target <- function(theta){
              w*dnorm(theta,mu1,sig1)+(1-w)*dnorm(theta,mu2,sig2)
          }
# Candidate density: N(0,3^2)
muC <- 0; sigC <- 3
# Initial values
theta0 <- (mu1+mu2)/2
burnIn <- 5000;
nSim <- 5000
set.seed(1)
# Generating Random Numbers
zBurn <- rnorm(burnIn, muC, sigC)
uBurn <- runif(burnIn)
zSim <- rnorm(nSim, muC, sigC)
uSim <- runif(nSim)
## Burn-In Step
thetaOld <- theta0
for (ii in 1:burnIn){
  thetaCand <- thetaOld + (muC+zBurn[ii]*sigC)
  alpha <- min(1,Target(thetaCand)/Target(thetaOld))
  if (uBurn[ii] <= alpha) thetaOld <- thetaCand
}
## MH samples
MHrs <- rep(0,nSim)
for (jj in 1:nSim){
  thetaCand <- thetaOld + (muC+zSim[jj]*sigC)
  alpha <- min(1,Target(thetaCand)/Target(thetaOld))
  if (uSim[jj] <= alpha) thetaOld <- thetaCand
  MHrs[jj] <- thetaOld
}

# Plotting
# install.packages("ggplot2")
library(ggplot2)
setEPS()
plot.new()
postscript('MHsamplerGaussianMixture101R.eps')  # Start to save figure
GMtrue <- p*dnorm(MHrs,mu1,sig1) + (1-p)*dnorm(MHrs,mu2,sig2)
RVdata1 <- data.frame(MHrs,GMtrue)
ggplot(RVdata1,aes(x=MHrs)) +
 geom_histogram(aes(y = ..density..),binwidth=0.5,fill="white",col="blue") +
 xlim(-7,22) +
 geom_line(aes(x=MHrs,y=GMtrue),lwd=1.2,col="red")
dev.off()            # End to save figure
# ------------------------------------------------------------------------------
```

이 R프로그램을 실행하면, 혼합정규확률밀도함수 (1)에서 번인기간이 5000이고 샘플들의 개수는 5000인 MH샘플들을 생성한다. 생성된 샘플들의 히스토그램과 진짜 혼합정규확률밀도함수가 그림 3.5.5에 그려져 있다. ■

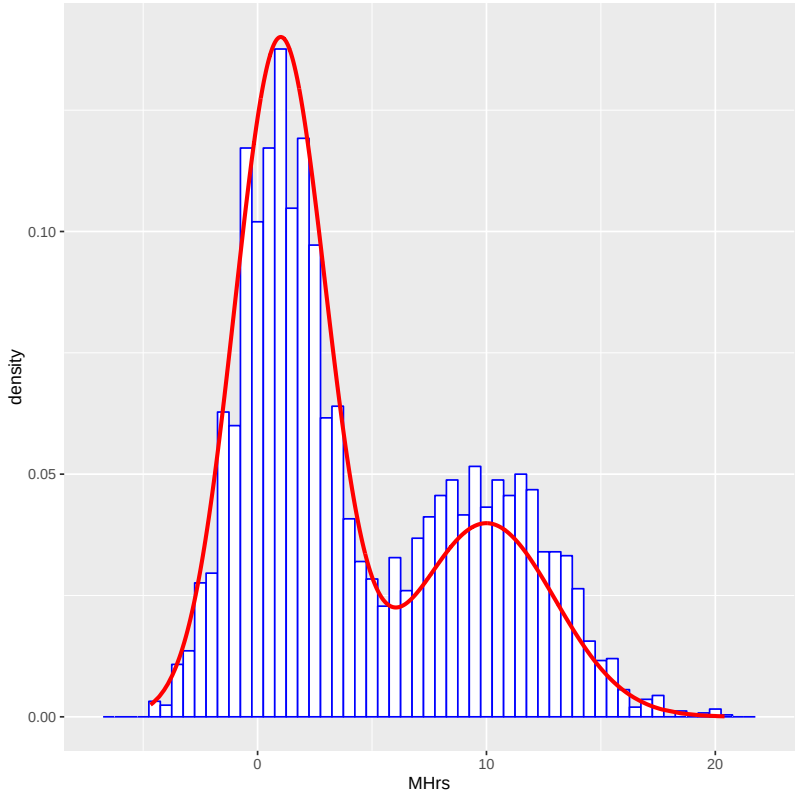


그림 3.5.5. MH샘플러에 의해 생성된 혼합정규샘플들의 히스토그램

제 4 장

준난수와 준몬테카를로법

이 절에서는 준난수(quasi-random number)와 준몬테카를로법(quasi-Monte Carlo method)에 대해서 다룰 것이다. 이에 대한 좀 더 자세한 내용은 Glasserman [17]을 참조하라.

제4.1절 준난수

몬테카를로법은 의사난수를 사용하여 적분을 계산하는 방법이다. 이에 반해, 준몬테카를로법은 준난수를 사용해서, d차원 초입방체 $[0,1]^d$에서 적분을 계산하는 방법이다. 준난수를 저불일치점열(low-discrepancy sequence)이라 부르기도 한다. 만약 풀고자 하는 문제를 $[0,1]^d$에서 적분으로 표현할 수 있다면, 준난수를 사용하는 준몬테카를로법을 적용할 수 있다. 준몬테카를로법의 계산절차는 몬테카를로법과 다를바 없다. 만약 몬테카를로법의 프로그램이 있다면, 의사난수를 준난수로 바꾸기만 하면 된다. 그러나, 몬테카를로법과 준몬테카를로법은 분명한 차이가 있다. 첫째, 이용하는 점열의 성질이 다르고, 둘째, 기반이 되는 정리가 다르며, 셋째 오차의 차수가 다르다. 대수법칙과 중심극한정리에서 알 수 있듯이, 몬테카를로적분값은 시행횟수 n에 대해 오차 $O(n^{-1/2})$로 구하고자 하는 적분값에 수렴한다. 반면에, 뒤에서 다루게 될 Koksma-Hlawka의 부등식에 의해서 알 수 있듯이, 준몬카를로적분값은 오차 $O(n^{-1}[\log n]^d)$로 적분값에 수렴한다. 따라서, 준몬테카를로법을 사용하면, 몬테카를로법보다 적은 시행횟수로 적분값에 가까운 근사값을 얻을 수 있다. 결과적으로, 금융업계의 프론트오피스와 같이 계산시간에 대해 제약을 받는 분야에서 준몬테카를로법이 주목받게 되었다.

우선, 1차원 정적분 $\theta \doteq \int_0^1 f(x)dx$를 수치적으로 계산하기로 하고, 다음 식들을 만족하는 적분구간 $[0,1]$의 분할 $\Pi \doteq \{x_0 < x_1 < \cdots < x_n\}$을 정의하자.

$$x_j \doteq \frac{j}{n}, \quad (j = 0, 1, \cdots, n) \tag{4.1.1}$$

이 분할을 사용해서 정적분 $\int_0^1 f(x)dx$를 다음과 같이 추정하기로 하자.

$$\hat{\theta}_{NA} \doteq \frac{1}{n}\sum_{j=0}^{n-1} f(x_j) \tag{4.1.2}$$

여기서 NA는 수치해석(numerical analysis)을 뜻한다. 만약 $f \in C^1$이면, 다음 식이 성립한다.

$$\left| \int_{x_j}^{x_{j+1}} f(x)dx - \frac{1}{n} f(x_j) \right| \leq \frac{1}{n^2} M \tag{4.1.3}$$

여기서 $M \doteq \sup_{x\in(0,1)} \{|f'(x)|\}$이다. 식 (4.1.3)에서 알 수 있듯이, 다음 식이 성립한다.

$$\left| \int_0^1 f(x)dx - \frac{1}{n}\sum_{j=0}^{n-1} f(x_j) \right| \leq \frac{1}{n} M \tag{4.1.4}$$

이 수치적분의 오차는 다음 식을 만족한다.

$$\epsilon_n \doteq \left| \int_0^1 f(x)dx - \frac{1}{n}\sum_{j=0}^{n-1} f(x_j) \right| = O\left(\frac{1}{n}\right) \tag{4.1.5}$$

만약 함수 f가 C^2급이거나 C^3급이면, 좀 더 높은 정밀도를 갖는 수치적분법을 사용할 수 있다. 예를 들어, Gauss구적법을 사용하면 아주 높은 정밀도를 갖는 수치적분값을 구할 수 있다. 몬테카를로법(Monte Carlo method)을 사용한 정적분 $\theta = \int_0^1 f(x)dx$의 추정값은 다음과 같다.

$$\hat{\theta}_{MC} \doteq \frac{1}{n}\sum_{j=1}^{n} f(u_j) \tag{4.1.6}$$

여기서 $\{u_j\}$는 일양난수들이다. 함수 f가 평활한 함수라고 가정하면, $\hat{\theta}_{MC}$는 불편추정값이고 분산이 $O(n^{-1})$임을 알 수 있다. 확률변수열 $\{y_n\}$이 식 $Var(y_n) = O(n^{-2\alpha})$을 만족하면, 이 확률변수열은 확률적으로(in probability) 오차의 차수가 $n^{-\alpha}$라고 하고 다음과 같이 표기한다.

$$y_n = O_P(n^{-\alpha}) \tag{4.1.7}$$

여기서 유의할 점은 결정적 차수 O는 최악의 시나리오를 상정하지만, 확률적 차수 O_P는 오차의 확률적 행태라는 것이다. 몬테카를로추정량 $\hat{\theta}_{MC}$는 다음 식을 만족한다.

$$\hat{\theta}_{MC} = \theta + O_P\left(\frac{1}{\sqrt{n}}\right) \tag{4.1.8}$$

식 (4.1.5)와 식 (4.1.8)에서 알 수 있듯이, 몬테카를로법에 의한 오차보다는 수치적분에 의한 오차가 작다. 이러한 이유로 1차원 정적분을 구하는 데는 몬테카를로법보다 수치적분법이 선호된다.

다음으로, 2차원 정적분 $\eta \doteq \int_0^1 \int_0^1 f(x_1, x_2) dx_1 x_2$를 계산하는 문제를 생각해보자. 편의상 n이 자연수 m의 제곱이라고 가정하자. 즉, $n = m^2$라고 하자. 다음과 같은 수열 $\{x_j\}$를 정의하자.

$$x_j \doteq \frac{j}{m}, \quad (j = 0, 1, \cdots, m) \tag{4.1.9}$$

다음 식을 만족하는 상수 $M(< \infty)$이 존재한다.

$$\left| \int_0^1 \int_0^1 f(x_1, x_2) dx_1 x_2 - \frac{1}{m^2} \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} f(x_i, x_j) \right| \leq \frac{1}{m^2} M = \frac{1}{n} M \tag{4.1.10}$$

식 (4.1.10)에서 알 수 있듯이, 다음 식들이 성립한다.

$$\hat{\eta}_{NA} \doteq \frac{1}{m^2} \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} f(x_i, x_j) = \eta + O\left(\frac{1}{m}\right) = \eta + O\left(\frac{1}{\sqrt{n}}\right) \tag{4.1.11}$$

몬테카를로법을 사용한 정적분 η의 추정값은 다음과 같다.

$$\hat{\eta}_{MC} \doteq \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} f(u_i, u_j) \tag{4.1.12}$$

여기서 $\{u_j\}$는 일양난수들이다. 함수 f가 평활한 함수라고 가정하면, $\hat{\eta}_{MC}$는 불편추정량이고 분산이 $O(m^{-2})$임을 알 수 있다. 따라서, 몬테카를로추정량 $\hat{\eta}_{MC}$를 다음과 같이 표기할 수 있다.

$$\hat{\eta}_{MC} = \eta + O_P\left(\frac{1}{m}\right) = \eta + O_P\left(\frac{1}{\sqrt{n}}\right) \tag{4.1.13}$$

식 (4.1.11)과 식 (4.1.13)에서 알 수 있듯이, 격자 상에서 평균값을 계산하는 수치적분에 의한 결정적 오차 $O(n^{-1/2})$는 몬테카를로법에 의한 확률적 오차 $O_P(n^{-1/2})$와 차원이 비슷하다. 따라서, 2차원인 경우에는 수치적분법과 몬테카를로법은 서로에게 경쟁상대가 된다.

다음과 같은 k차원 정적분을 계산해보자.

$$\psi \doteq \int_0^1 \int_0^1 \cdots \int_0^1 f(x_1, x_2, \cdots, x_k) dx_1 dx_2 \cdots dx_k \tag{4.1.14}$$

여기서 차수 k는 3이상이고 함수 f는 평활하다고 하자. 이 정적분을 다음과 같이 추정하기로 하자.

$$\hat{\psi}_{NA} \doteq \sum_{j_1=0}^{m}\sum_{j_2=0}^{m}\cdots\sum_{j_k=0}^{m} w_{j_1}w_{j_2}\cdots w_{j_k} f\left(\frac{j_1}{m},\frac{j_2}{m},\cdots,\frac{j_k}{m}\right) \tag{4.1.15}$$

여기서 $w_{j_1}, w_{j_2}, \cdots, w_{j_k}$는 사다리꼴공식에 적합한 가중값들이다. 여기에 사용된 관찰점들은 $n = [m+1]^k$ 개이다. 따라서, k차원 정적분 ψ를 수치적분값 $\hat{\psi}_{NA}$로 추정하면, 그 오차는 다음과 같다.

$$\hat{\psi}_{NA} = \psi + O(m^{-2}) = \psi + O(n^{-2/k}) \tag{4.1.16}$$

즉, 오차의 차원은 지수적으로 증가한다. 따라서, 오차를 일정하게 유지하기 위해서는 k가 증가함에 따라 관찰점들의 개수를 지수적으로 늘려야 한다. 이러한 현상을 차수의 저주(curse of dimensionality)라 부른다. 반면에, 몬테카를로법은 이러한 차수문제로부터 자유롭다. 이 몬테카를로추정량 $\hat{\psi}_{MC}$가 불편이고 분산이 $O(m^{-k})$임을 알 수 있다. 따라서, 몬테카를로추정량 $\hat{\psi}_{MC}$를 다음 식을 만족한다.

$$\hat{\psi}_{MC} = \psi + O_P(m^{-k/2}) = \psi + O_P\left(\frac{1}{\sqrt{n}}\right) \tag{4.1.17}$$

식 (4.1.17)에서 알 수 있듯이, 몬테카를로법에 의한 오차는 차수 k에 의존하지 않는다. 이러한 장점은 적분영역이 복잡한 경우에도 성립한다. 식 (4.1.16)과 식 (4.1.17)에서 알 수 있듯이, 고차원 정적분에서는 몬테카를로법에 의한 확률적 오차 $O_P(n^{-1/2})$가 격자 상에서 평균값을 계산하는 수치적분에 의한 결정적 오차 $O(n^{-2/k})$보다 작음을 알 수 있다. 실제로, 격자 상에서 평균값으로 계산되는 수치적분값은 최적과는 거리가 멀고, 의사난수(psuedo-random number)를 이용하는 몬테카를로법이나 준난수(quasi-random number)를 이용하는 준몬테카를로법이 수치적분을 사용하는 것보다 훨씬 더 효율적이다. 그렇다고 해서 몬테카를로법이나 준몬테카를로법이 수치적분에 비해 절대적으로 우위에 있다는 것은 아니다. 몬테카를로법의 첫 번째 약점은 결정적이 아닌 확률적 오차범위 만을 제시할 수 있다는 것이다. 두 번째 약점은 피적분함수의 정칙성(regularity)이 알려진 경우에도 이를 이용하지 않는다는 것이다. 몬테카를로법의 오차 $O_P(n^{-1/2})$는 매우 약한 정칙성조건 하에서도 성립한다. 그러나, 피적분함수의 추가적 정칙성이나 평활에 관한 정보가 주어진다 해도 몬테카를로법에서는 이를 사용할 수 없다. 세 번째 약점은 진짜 몬테카를로법을 적용할 수 없고 대신 의사난수를 사용하는 시뮬레이션으로 대체한다는 것이다.

준난수는 정적분을 하기 위한 것이지, 난수 자체를 연구하기 위한 것이 아니다. 따라서,

준난수는 주변확률분포함수가 일양성(uniformity)을 갖는다는 검정을 제외한 의사난수에 관한 어떤 검정도 적용하지 못한다. 그 이유는 준난수들은 구조상 자기상관성을 갖기 때문이다. 이러한 이유로 준몬테카를로법을 다룰 때에는 '일양으로' 대신 '균등하게' 라는 단어를 사용하고자 한다. 우리의 목표는 n개 점들에서 함수값들을 구한 다음 이들의 표본평균으로 정적분을 구하는 것이다. 따라서, 이 n개 점들이 적분영역에서 골고루 퍼져있으면 정적분의 좋은 근사값을 얻을 수 있을 것이다.

제4.2절 저불일치점열

구간 $[0,1]$에 속하는 n개 점들로 이루어진 점집합 $P = \{x_1, x_2, \cdots, x_n\}$을 생각해보자. 고정된 점 $y \in (0,1]$에 대해서 집합 $B(y) \doteq [0,y)$에 들어가는 점집합 P의 원소들의 개수를 $A(B(y);P)$로 표기하고, 점집합 P의 디스크레펀시(discrepancy)를 정의하기로 하자. 첫째, 집합 $B(y) = [0,y)$에 들어가는 점의 비율을 $z(y)$로 표기하자. 다음 식이 성립한다.

$$z(y) \doteq \frac{A(B(y);P)}{n} \tag{4.2.1}$$

이 $z(y)$는 일양확률분포(uniform probability distribution)에서 추출된 샘플들 $x_1, x_2, \cdots, x_n$의 경험확률분포함수이다. 따라서 다음 식이 성립한다.

$$z(y) = \frac{1}{n}\sum_{j=1}^{n} 1(x_j \le y) \tag{4.2.2}$$

둘째, 함수 $f = \{[y,z] \mid 0 \le y \le 1\}$의 궤적을 평면에 그리면, 계단함수가 그려진다. 셋째, 이 평면에 추가로 직선 $g = \{[y,z] \mid z = y, 0 \le y \le 1\}$를 그린다. 넷째, 이 계단함수와 직선의 차이 정도를 점집합 P의 디스크레펀시라 부른다. 즉, 디스크레펀시는 두 함수들 f와 g의 거리로 측정된다.

예제 4.2.1 점집합 $P = \{x_1, x_2, \cdots, x_5\}$에 대한 디스크레펀시를 그리기 위해서는 다음 MATLAB파일 Discrepancy101.m을 실행하라.

```
% ------------------------------------------------------------------------
% Filename: Discrepancy101.m
% Definition of Discrepancy
% Programmed by CBS
% ------------------------------------------------------------------------
clear, clf, close all
n = 5
```

```
xx1 = [ 0 1/2/n:1/n:(1-1/2/n) 1 ]
ff = [ 0:1/n:1 1 ]
rand('twister',5489)
xx2 = [ 0 sort(rand(1,n)) 1 ]
subplot(1,2,1)
stairs(xx1,ff,'k-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
hold on
plot([ 0 1 ],[ 0 1 ],'r--','linewidth',2)
legend('\bf f','\bf g','location','SE')
plot([ 0 0],[-0.2 1.2],'b-',[-0.2 1.2],[ 0 0],'b-','linewidth',1.5)
axis([-0.1 1.1 -0.1 1.1 ])
hold off
subplot(1,2,2)
stairs(xx2,ff,'k-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
hold on
plot([ 0 1 ],[ 0 1 ],'r--','linewidth',2)
legend('\bf f','\bf g','location','SE')
plot([ 0 0],[-0.2 1.2],'b-',[-0.2 1.2],[ 0 0],'b-','linewidth',1.5)
axis([-0.1 1.1 -0.1 1.1 ])
hold off
saveas(gcf,'Discrepancy101','epsc')
save('Discrepancy101','xx1','xx2')
% End of program
% --------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 그림 4.2.1이 그려진다. 그림 4.2.1의 좌측 그래프에서는 집합 P의 다섯 점들이 균등하게 분포되어 있고, 따라서 f와 g의 편차는 작으며, 결과적으로 점집합 P의 디스크레펀시는 작다. 반면에, 그림 4.2.1의 우측 그래프에서는 집합 P의 점들이 오른쪽으로 몰려 있으며, 따라서 f와 g의 편차는 크고, 결과적으로 점집합 P의 디스크레펀시는 크다. ■

경험확률분포함수 $z(y)$가 일양확률분포함수에 얼마나 가까운가를 나타내는 다양한 척도들이 있다. 경험확률분포함수와 진짜 확률분포함수가 유의적으로 같은가를 검정하는 Kolmogorov-Smirnov검정법을 바탕으로, 다음과 같은 척도를 정의하기로 하자.

정의 4.2.1

점집합 $P = \{x_1, x_2, \cdots, x_n\}$에 대해서 다음과 같이 정의되는 $D_n^*(P)$를 점집합 P의 스타디스크레펀시(star discrepancy)라 부른다.

$$D_n^*(P) \doteq \sup_{y\in[0,1]} \left| \frac{A(B(y);P)}{n} - y \right|$$

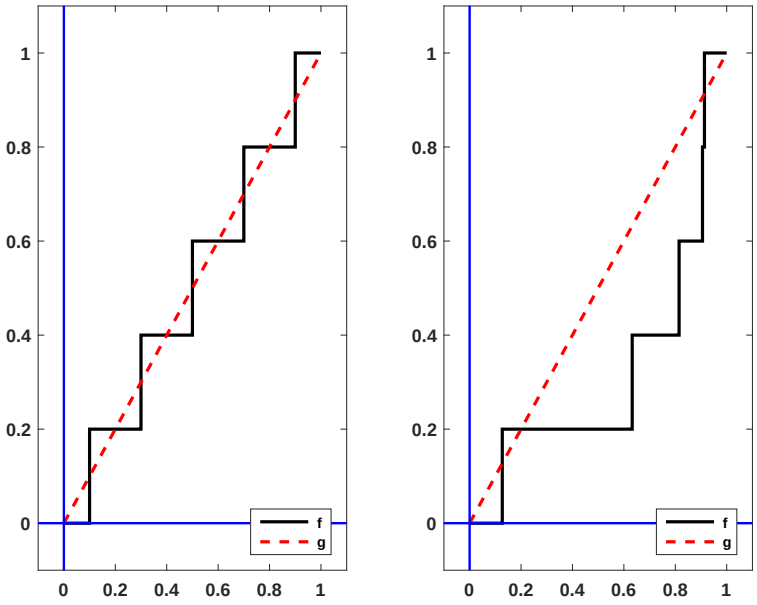


그림 4.2.1. 디스크레펀시

직관적으로 보아도, 각 n에 대해서 디스크레펀시가 작은 점들 $x_1, x_2, \cdots, x_n$을 택하는 것이 좋을 것이다. 즉, 스타디스크레펀시 $D_n^*(P)$를 작게 하는 점집합 P를 선택함으로써 정적분의 좋은 근사값을 얻을 수 있을 것이다. 그렇다면, $D_n^*(P)$가 얼마면 작다고 할 것인가? 이에 대한 답으로 다음과 같은 정의를 하자.

정의 4.2.2

각 $n(\geq 2)$에 대해서 다음 부등식을 만족하는 점열 P를 저불일치점열(低不一致点列, low-discrepancy sequence)이라 부른다.

$$D_n^*(P) \leq c\frac{\log n}{n}$$

여기서 c는 상수이다.

피적분함수 f가 평활하면, 저불일치점열에 의한 정적분은 참값에 가깝다. 이에 대해서 자세히 기술하려면, 먼저 유계변분(bounded variation)에 대해서 정의해야 한다. 함수 $f(\in C^1[0,1])$에 대해서 Hardy-Krause변분 $V(f)$를 다음과 같이 정의한다.

$$V(f) \doteq \int_0^1 |f'(x)|dx \tag{4.2.3}$$

만약 $V(f)$가 유한이면, 함수 f는 구간 $[0,1]$에서 유계변분을 갖는다고 한다. 저불일치점열을 사용한 준몬테카를로법에 의한 오차는 다음과 같다.

정리 4.2.1: Koksma부등식

함수 f의 구간 $[0,1]$에서 변분 $V(f)$가 유계이면, 점집합 $P = \{x_1, x_2, \cdots, x_n\}$에 대해서 다음 식이 성립한다.

$$\left| \int_0^1 f(x)dx - \frac{1}{n}\sum_{j=1}^{n} f(x_n) \right| \leq V(f)D_n^*(P)$$

정리 4.2.1 에서 알 수 있듯이, 준몬테카를로법에 의한 근사값과 진짜 정적분의 차이는 유계변분 $V(f)$와 스타디스크레펀시 $D_n^*(P)$의 곱 $V(f)D_n^*(P)$를 넘지 않는다.

지금부터는 d차원 디스크레펀시에 대해서 살펴보자. 다음과 같은 점집합을 정의하자.

$$P \doteq \left\{ \boldsymbol{x}_j = \left[x_j^{(2)}, x_j^{(2)}, \cdots, x_j^{(d)} \right] \mid j = 1, 2, \cdots, n \right\} \ (\in [0,1]^d) \tag{4.2.4}$$

또한, 점 $\boldsymbol{y} = \left[y^{(1)}, y^{(2)}, \cdots, y^{(d)} \right]$ $(\in [0,1]^d)$에 대해서 다음과 같은 집합을 정의하자.

$$J(\boldsymbol{y}) \doteq \left[0, y^{(1)}\right) \times \left[0, y^{(2)}\right) \times \cdots \times \left[0, y^{(d)}\right) \ \left(\subset [0,1]^d\right) \tag{4.2.5}$$

집합 $J(\boldsymbol{y})$에 속하는 집합 P의 원소들의 개수를 $A(J(\boldsymbol{y}); P)$로 표기하자.

정의 4.2.3

식 (4.2.4)에서 정의된 점집합 P에 대해서 다음과 같이 정의되는 $D_n^*(P)$를 점집합 P의 스타디스크레펀시라 부른다.

$$D_n^*(P) \doteq \sup_{\boldsymbol{y} \in [0,1]^d} \left| \frac{A(B(\boldsymbol{y}); P)}{n} - \prod_{j=1}^{d} y^{(j)} \right|$$

다차원 저불일치점열은 1차원과 비슷하게 정의된다. 다차원 저불일치점열에는 Halton열, Sobol열, Faure열 등이 있다.

정의 4.2.4

각 $n(\geq 2)$에 대해서 다음 부등식을 만족하는 점열 P를 저불일치점열이라 부른다.

$$D_n^*(P) \leq c\frac{[\log n]^d}{n}$$

여기서 c는 상수이다.

정의역이 초입방체(unit hypercube) $[0,1]^d$이고 적절한 차원까지 미분가능한 함수 f에 대해서 Hardy-Krause변분 $V(f)$을 정의하기 위해서, 먼저 다음과 같은 초사각형(hyperrectangle 또는 orthotope) $J_{\boldsymbol{u}}$를 정의하자.

$$J_{\boldsymbol{u}} \doteq \left[u_1^-, u_1^+\right] \times \left[u_2^-, u_2^+\right] \times \cdots \left[u_d^-, u_d^+\right] \tag{4.2.6}$$

여기서 각 $i(=1,2,\cdots,d)$에 대해서 부등식들 $0 \leq u_i^- \leq u_i^+ \leq 1$이 성립한다고 하자. 이 초사각형의 정점의 각 좌표는 u_i^- 또는 u_i^+이다. 이 정점들 중에서 u_i^+가 짝수개인 집합을 E_J 그리고 홀수개인 집합을 O_J로 표기하고, 다음 함수를 정의하자.

$$\Delta\left(f;J\right) \doteq \sum_{\boldsymbol{u}\in E_J} f(\boldsymbol{u}) - \sum_{\boldsymbol{u}\in O_J} f(\boldsymbol{u}) \tag{4.2.7}$$

초입방체 $[0,1]^d$를 J_u 형태를 갖는 초사각형들로 분할한 집합을 P_J라고 하고, 다음과 같이 함수 f의 변분의 측도를 정의하자.

$$V^{(d)}(f) = \sup_{P_J} \sum_{J_{\boldsymbol{u}}} |\Delta(f;J_{\boldsymbol{u}})| \tag{4.2.8}$$

만약 해당하는 편도함수들이 미분가능이면, 다음 식이 성립함을 증명할 수 있다.

$$V^{(d)}(f) = \int_0^1 \cdots \int_0^1 \left|\frac{\partial^d f}{\partial u_1 \cdots \partial u_d}\right| du_1 \cdots du_d \tag{4.2.9}$$

초입방체 $[0,1]^d$에서 정의되는 함수 f의 정의역을 다음과 같이 제한하는 함수를 정의하자. 임의의 $k(=1,2,\cdots,d)$와 임의의 $1 \leq i_1 < i_2 < \cdots < i_k \leq d$에 대해서, 만약 $l \notin \{i_1, i_2, \cdots, i_k\}$이면, $u_l = 1$인 점 $[u_1, u_2, \cdots, u_d]$에서 함수 f를 함수 $f_{i_1,i_2,\cdots,i_k}$로 표기하자. 또한, 다음과

같이 함수 $V^{(k)}\left(f;i_1,i_2,\cdots,i_k\right)$를 정의하자.

$$V^{(k)}\left(f;i_1,i_2,\cdots,i_k\right) \doteq V^{(k)}\left(f_{i_1,i_2,\cdots,i_k}\right) \tag{4.2.10}$$

함수 $V^{(k)}\left(f;i_1,i_2,\cdots,i_k\right)$를 사용해서 함수 f의 Hardy-Krause변분을 다음과 같이 정의하자.

$$V(f) \doteq \sum_{k=1}^{d} \sum_{1\le i_1<\cdots<i_k\le d} V^{(k)}\left(f_{i_1,i_2,\cdots,i_k}\right) \tag{4.2.11}$$

만약 $V(f)$가 유한이면, 함수 f는 초입방체 $[0,1]^d$에서 유계변분을 갖는다고 한다.

고차원 저불일치점열을 사용한 준몬테카를로법에서 오차는 다음과 같다.

정리 4.2.2: Koksma-Hlawka부등식

함수 f가 $[0,1]^d$에서 Hardy-Krause의미에서 유계변분 $V(f)$를 가지면, 점집합 $P=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_n\}$에 대해서 다음 식이 성립한다.

$$\left|\int_{[0,1]^d} f(\boldsymbol{x})d\boldsymbol{x} - \frac{1}{n}\sum_{j=1}^{n} f\left(\boldsymbol{x}_n\right)\right| \le V(f)D_n^*(P)$$

실제로는 정리 4.2.2의 오차상한 $V(f)D_n^*(P)$를 잘 사용하지 않는다. 그 이유는 Hardy-Krause변분 $V(f)$를 구하는 것이 매우 어렵기 때문이다. 그러나, 이 오차상한 $V(f)D_n^*(P)$는 피적분함수 자체의 성질인 변분 $V(f)$와 점열 P 자체의 성질인 스타디스크레펀시 $D_n^*(P)$를 분리하므로, $D_n^*(P)$가 작으면 준몬테카를로법에 의한 오차가 작음을 알 수 있다. 따라서, 준몬테카를로법에서 디스크레펀시는 매우 중요한 것이다. 정의 4.2.4에서 알 수 있듯이, 저불일치점열에서는 부등식 $D_n^*(P) \le cn^{-1}[\log n]^d$가 성립하므로, 준몬테카를로법에 의한 오차의 차수는 시행횟수 n에 대해 $O\left(n^{-1}[\log n]^d\right)$가 된다. 반면에, 의사난수를 사용한 몬테카를로법에 의한 오차는 점근적으로 정규확률분포에 따르며, 오차의 차수는 시행횟수 n에 대해 $O\left(n^{-1/2}\right)$이다. 따라서, 정밀도를 1자리수만큼 올리기 위해서는, 즉 오차를 1/10으로 만들기 위해서, 몬테카를로법에서는 시행횟수 n을 100배로 할 필요가 있다. 그러나, 준몬테카를로법에서는 수십배 정도이면 된다. 즉, 저불일치점열을 사용하면, 의사난수를 사용한 경우에 비해 작은 n으로 필요한 정밀도를 얻을 수 있다. 또한, 몬테카를로법에서 오차평가는 중심극한정리을 바탕으로 하는 확률적 평가이므로, 오차가 아주 큰 확률이 0에 가깝다고는 할 수 있지만 오차의 상한이 반드시 존재하는 것은 아니다. 반면에, 준몬테카를로법에서는

Koksma-Hlawka부등식에 의해 오차의 확정적인 상한을 얻을 수 있다. Hardy-Krause변분과 Koksma-Hlawka부등식에 대해서는 Glasserman [17, 제5.1.3소절]을 참조하라.

제4.3절 Van der Corput 열

Van der Corput 열은 1차원 저불일치점열이다. 이 Van der Corput 열을 사용해서, 1차원 구간 $[0,1]$에서 정적분 $\int_0^1 f(u)du$를 계산하는 방법을 살펴보기로 하자.

고정된 n에 대해서 다음과 같은 구간 $[0,1]$의 분할을 살펴보자.

$$0 = x_0 < x_1 < \cdots < x_{n-1} < x_n = 1 \tag{4.3.1}$$

이 $\{x_j\}$를 사용해서 다음과 같이 정적분의 근사값을 계산한다.

$$\theta \doteq \int_0^1 f(u)du \approx \frac{1}{n}\sum_{j=0}^{n} w_j f(x_j) \tag{4.3.2}$$

여기서 j가 0이거나 n인 경우에는 가중값 w_j가 1/2이고, 다른 경우에는 가중값 w_j가 1이다. 즉, 식 (4.3.2)의 우변은 사다리꼴공식을 사용해서, 정적분 θ의 근사값을 구한 것이다. 다음 예제는 식 (4.3.2)의 우변을 좀 더 체계적으로 계산하기 위한 것이다.

예제 4.3.1 다음과 같이 $\{x_j \mid j = 0, 1, \cdots, n(=2^m)\}$를 선택하기로 하자.

첫째, x_0로 0을 선택한다.

둘째, x_1을 구간 $[0,1]$의 중점 0.5로 한다.

셋째, x_2를 소구간 $[0, 0.5]$의 중점 0.25로 한다.

넷째, x_3를 소구간 $[0.5, 1]$의 중점 0.75로 한다.

다섯째, x_4를 소구간 $[0, 0.25]$의 중점 0.125로 한다.

여섯째, x_5를 소구간 $[0.5, 0.725]$의 중점 0.625로 한다.

일곱째, x_6를 소구간 $[0.25, 0.5]$의 중점 0.375로 한다.

여덟째, x_7을 소구간 $[0.75, 1]$의 중점 0.875로 한다.

같은 방법으로, 바로 전 단계의 중점들의 점집합에 0과 1을 추가한 다음, 그 점들의 중점들의 점집합을 구한다. 마지막으로 x_n에 1을 할당한다.

이 점집합을 그리기 위해서, 다음 MATLAB파일 MidpointSequence101.m을 실행해보자.

```
% ------------------------------------------------------------------
```

```
%  Filename: MidpointSequence101.m
%  Generating Midpoint Sequence
%  Programmed by CBS
%--------------------------------------------------------------------------
clear all, close all
k = 5
x = [ 0 ];
plot(x,0,'k.','linewidth',3)
set(gca,'fontsize',11,'fontweigh','bold','xtick',0:1/8:1)
hold on
plot(x(end),0,'ro','linewidth',1.5)
for kk=1:1:k
    xdum1 = [ x(2:end), 1 ];
    xdum2 = (x+xdum1)/2;
    plot(xdum2,kk,'ro','linewidth',1.5)
    x = [ x xdum2 ];
    x = sort(x);
    x = unique(x)
    plot(x,kk,'k.','linewidth',3)
end
axis([ -0.1 1.1 -0.5 kk+0.5 ])
xlabel('\bf x'), ylabel('\bf k','rotation',0)
hold off
saveas(gcf,'MidpointSequence101','epsc')
save('MidpointSequence101','x')
% End of program
% --------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 그림 4.3.1이 그려진다. 그림 4.3.1에서 알 수 있듯이, 이렇게 구한 점집합은 구간 $[0,1]$을 등간격으로 나누는 분할이다. 예를 들어, $n = 2^5$ 인 경우에는 구간 $[0,1]$을 32등분하는 점집합이 얻어진다. 그림 4.3.1의 각 행에서 흑색 다이아몬드는 기존의 점들을 나타내고, 적색 원으로 둘러쌓인 다이아몬드는 새로운 점들을 나타낸다. ■

예제 4.3.1의 방법을 사용하면, $n = 2^m$ 인 경우에는 등간격인 점들로 구성된 점열 $\{x_j \mid j = 0, 1, \cdots, n(= 2^m)\}$를 생성한다. 그러나, 만약 n이 2의 지수형태가 아니면, 이 점집합은 구간 $[0,1]$중에서 0에 가까운 점들을 더 많이 포함할 것이다. 지금부터는 $n = 2^m$가 아닌 경우에도 가능한 한 점들이 균등하게(uniformly) 분포하게 하는 방법을 생각해보자. 이러한 목적을 달성하기 위해서, 근기역함수(**根基逆函數**, radical-inverse function)와 Van der Corput열을 정의하자.

정의 4.3.1

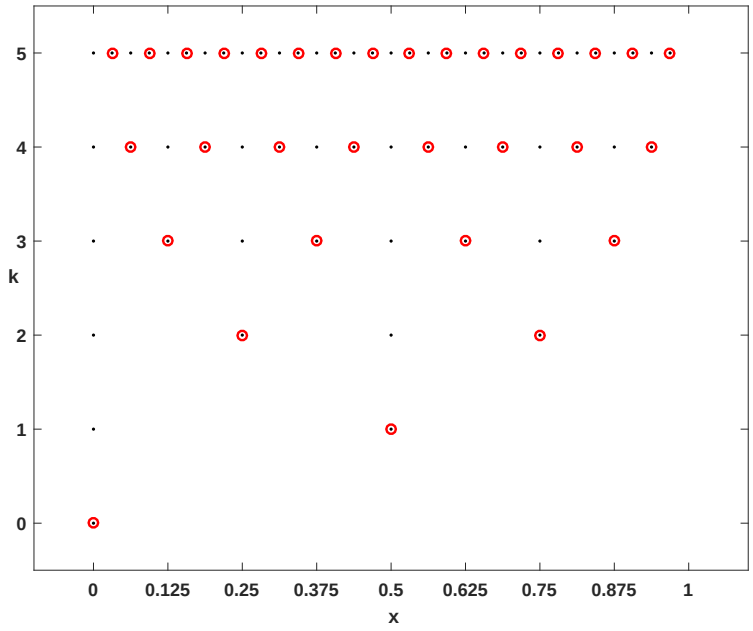


그림 4.3.1. 중심점열

정수 $n(\geq 0)$를 b진법으로 나타낸 b^l 항의 계수를 $a_l(n)$이라 하면, 다음 식이 성립한다.

$$n = \sum_{l=0}^{\infty} a_l(n) b^l$$

즉, $n = (\cdots a_3(n)a_2(n)a_1(n)a_0(n))_b$이다. 이에 대해서 다음과 같은 함수를 정의하자.

$$\psi_b(n) \doteq \sum_{l=0}^{\infty} a_l(n) b^{-l-1}$$

즉, $\psi_b(n) = (0.a_0(n)a_1(n)a_2(n)a_3(n)\cdots)_b$이다. 이 함수 $\psi_b(\cdot)$를 기수(**基數**, base number)가 b인 근기역함수라 한다.

정의 4.3.1에서 알 수 있듯이, b진법으로 나타낸 $\psi_b(n)$은 b진법으로 나타낸 n을 소수점에서 대칭으로 접은 것이다. 따라서, 각 정수 $n(\geq 0)$에 대해서 $\psi_b(n)$은 구간 $[0, 1)$에 속한다. 여기서 유의할 점은 각 자연수 n에 대해서 $r(n) \doteq \lfloor \ln n / \ln b + 1 \rfloor$보다 큰 l에 대해서 식 $a_l(n) = 0$가 성립한다는 것이다. 즉, n을 다음과 같이 표현할 수 있다.

$$n = \sum_{l=0}^{r(n)-1} a_l(n) b^l \tag{4.3.3}$$

정의 4.3.2: Van der Corput 열

자연수 $b(\geq 2)$에 관해 점열 $\{\psi_b(n) \mid n = 0, 1, \cdots\}$을 기수가 b인 Van der Corput 열이라 한다.

예제 4.3.2 정수 $0, 1, 2, 3, 4$를 2진수로 나타내면 각각 $0_2, 1_2, 10_2, 11_2, 100_2$이므로, 기수가 2인 근기역함수는 다음과 같다.

$$\psi_2(0) = 0_2, \;\; \psi_2(1) = 0.1_2, \;\; \psi_2(2) = 0.01_2, \;\; \psi_2(3) = 0.11_2, \;\; \psi_2(4) = 0.001_2 \tag{1}$$

따라서, 2진수로 나타낸 기수가 2인 Van der Corput 열은 다음과 같다.

$$\{0_2, 0.1_2, 0.01_2, 0.11_2, 0.001_2, 0.101_2, 0.011_2, 0.111_2, \cdots\} \tag{2}$$

이를 10진수로 나타내면 다음과 같다.

$$\left\{0, \frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{8}, \frac{5}{8}, \frac{3}{8}, \frac{7}{8}, \ldots\right\} \tag{3}$$

같은 방법으로, 기수가 3인 근기역함수는 3진수로 다음과 같음을 알 수 있다.

$$\psi_3(0) = 0_3, \;\; \psi_3(1) = 0.1_3, \;\; \psi_3(2) = 0.2_3, \;\; \psi_3(3) = 0.01_3, \;\; \psi_3(4) = 0.11_3 \tag{4}$$

■

기수가 2인 Van der Corput 열 $\{x_n\}$을 소수점 이하 한자리수, 즉 2^{-1} 자리까지 자르면, 0.0과 0.1이 반복된다. 따라서, $\{x_n\}$의 연속한 두 점들 x_n과 x_{n+1}은 $[0, 0.5)$와 $[0.5, 1)$에 하나씩 들어간다. 즉, 어느 한쪽 구간에 연속으로 들어가는 경우는 없다. 마찬가지로 소수점 이하 두 자리, 즉 2^{-2} 자리까지 자르면, $0.00, 0.10, 0.01, 0.11$을 반복한다. 따라서 $\{x_n\}$의 연속한 4가지 점 $x_n, x_{n+1}, x_{n+2}, x_{n+3}$은 $[0, 0.25), [0.25, 0.5), [0.5, 0.75), [0.75, 1)$에 하나씩 들어간다. 동일한 이유로, 각 $j(= 3, 4, \cdots)$에 대해서 2^{-j} 자리까지 자르면, $\{x_n\}$의 연속한 2^j 개 점들은 $\left[0, 1 \cdot 2^{-j}\right)$, $\left[1 \cdot 2^{-j}, 2 \cdot 2^{-j}\right), \cdots,$ $\left[\left[2^{-j} - 2\right] \cdot 2^{-j}, \left[2^{-j} - 1\right] \cdot 2^{-j}\right)$, $\left[\left[2^{-j} - 1\right] \cdot 2^{-j}, 1\right)$에 하나씩 들어간다. 즉, 기수를 2로 하는 Van der Corput 열은 n이 2^k 형태가 아닌 경우에도 0의 부근에 점들이 많이 모이지 않고 골고루 퍼져있다. Van der Corput 열의 스타디스크레펀

시는 다음과 같다.

$$D_n^* = O\left(\frac{\log n}{n}\right) \tag{4.3.4}$$

예제 4.3.1에서는 구간 $[0,1]$의 양 끝점들 0과 1을 점열에 포함했다. 그러나, 예제 4.3.2에서는 구간 $[0,1]$의 한 끝점 0은 점열에 포함했으나 1은 포함하지 않았다. 일반적으로, 점열이 긴 경우에는 끝점들 0이나 1을 포함하는지 여부는 계산결과에 큰 영향을 끼치지 않는다. 그러나, 정의역이 $(-\infty,\infty)$인 확률분포에 역함수법을 적용할 때에는 주의가 필요하다. 예를 들어, 정규확률분포에서는 $N^{-1}(0) = -\infty$이고 $N^{-1}(1) = \infty$이므로, 실제 정적분을 계산할 때는 끝점들 0과 1을 제거할 필요가 있다.

예제 4.3.3 Van der Corput열을 생성하기 위해서, 다음 MATLAB프로그램 VanDerCorput101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: VanDerCorput101.m
%  Generating van der Corput Sequence
%  >> vdc = VanDerCorput101(18,2)
%  Programmed by CBS
%------------------------------------------------------------------------
function vdc = VanDerCorput101(n,b)
format rat
% Input: n = No of van der Corput numbers
r = floor(log(n)/log(b)+1);
nn = 1:n;
Amat = [ ];
for jj=0:r-1
    a = mod(nn,b);
    nn = (nn-a)/b;
    Amat = [ Amat; a ];
end
vdc = b.^(-(1:r))*Amat;
plot(vdc,'r-','linewidth',1.2)
hold on
set(gca,'fontsize',11,'fontweigh','bold','xlim',[1 n])
plot(vdc,'k*','linewidth',2)
xlabel('\bf k'), ylabel('\bf Van der Corput')
hold off
saveas(gcf,'VanDerCorput101','epsc')
save('VanDerCorput101','vdc')
end
% (Another Method) x = haltonset(1); vdc = net(x,n+1) for base=2
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 이용해서 기수가 2인 18개 Van der Corput점들을 생성하기 위해서 MATLAB커맨드행에서 다음 명령문을 실행하라.

```
>> vdc = VanDerCorput101(18,2)
```

여기서 MATLAB명령문 format rat는 답을 분수로 출력하기 위한 것이다. 이 MATLAB 명령문이 실행되면, 다음과 같은 결과를 얻는다.

```
vdc =
  Columns 1 through 6
      1/2         1/4         3/4         1/8         5/8         3/8
  Columns 7 through 12
      7/8         1/16        9/16        5/16        13/16       3/16
  Columns 13 through 15
     11/16        7/16        15/16       1/32        17/32       9/32
```

이 결과를 출력한 그래프가 그림 4.3.2에 수록되어 있다. ■

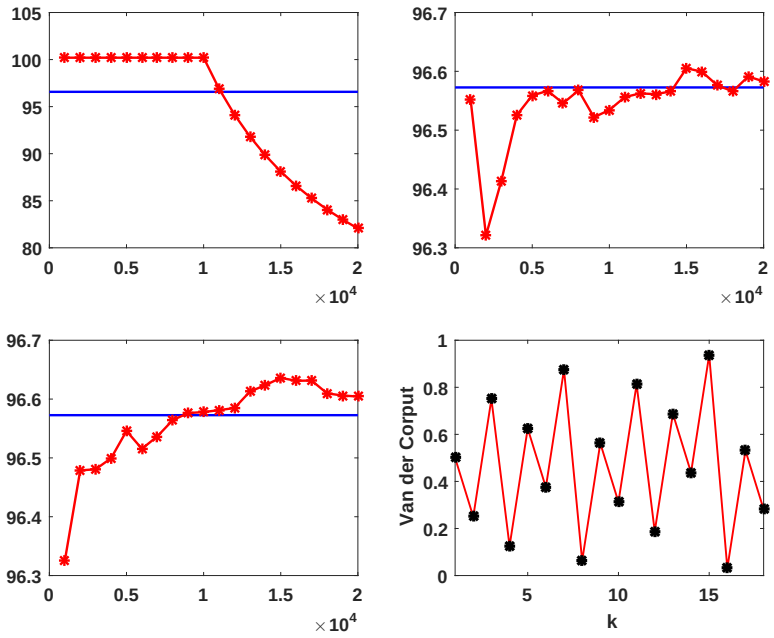


그림 4.3.2. Van der Corput열 1

자연수를 10진법에서 2진법으로 바꾸는 MATLAB함수 de2bi.m을 사용하면, 기수가 2인 Van der Corput점들을 쉽게 발생시킬 수 있다.

예제 4.3.4 기수가 2인 Van der Corput열을 생성하기 위해서, 다음 MATLAB프로그램 VanDerCorput102.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: VanDerCorputBase102.m
%  Generating van der Corput series with Base number b = 2
%  >> [ Amat vdc ] = VanDerCorputBase2(8)
```

```
%  Programmed by CBS
%--------------------------------------------------------------------
function [ Amat vdc ] = VanDerCorputBase102(n)
format rat
r = floor(log(n)/log(2)+1)
Amat = zeros(r,n);  % The k-th raw = [ a_0 (k) ... a_r-1 (k) ]
for k = 1:n
    dum = de2bi(k)';
    Amat(1:length(dum),k) = dum;
end
vdc = 2.^(-(1:r))*Amat;
% nn = 1:n;
plot(vdc,'r-','linewidth',1.2)
hold on
set(gca,'fontsize',11,'fontweigh','bold','xlim',[1 n])
plot(vdc,'k*','linewidth',1.2)
% plot(nn,vdc(1,:),'r*',nn,vdc(2,:),'g+',nn,vdc(1,:),'bd','linewidth',1.2)
hold off
saveas(gcf,'VanDerCorput102','epsc')
save('VanDerCorput102','vdc')
end
% End of program
% --------------------------------------------------------------------
```

이 MATLAB프로그램을 이용해서 기수가 2인 Van der Corput점들을 8개 생성하기 위해서 MATLAB커맨드행에서 다음 명령문을 실행하라.

```
>> [ Amat vdc ] = VanDerCorput102(8)
```

이 MATLAB명령문이 실행되면, 다음과 같은 결과를 출력한다.

```
Amat =

  Columns 1 through 8

     1     0     1     0     1     0     1     0
     0     1     1     0     0     1     1     0
     0     0     0     1     1     1     1     0
     0     0     0     0     0     0     0     1


vdc =

  Columns 1 through 8

   1/2   1/4   3/4   1/8   5/8   3/8   7/8  1/16
```

행렬 Amat의 제k열은 자연수 k를 2진법으로 나타내는 것이다. 즉, 행렬 Amat의 제k번 째 열벡터는 $[a_0(k), a_1(k), \cdots, a_{r-1}(k)]^t$이다. 따라서, 기수가 2인 근기역함수가 다음과

같음을 쉽게 알 수 있다.

$$\psi_2(1) = 0.1000_2,\ \psi_2(2) = 0.0100_2,\ \psi_2(3) = 0.1100_2,\ \psi_2(4) = 0.0010_2 \tag{1}$$

$$\psi_2(5) = 0.1010_2,\ \psi_2(6) = 0.0110_2,\ \psi_2(7) = 0.1110_2,\ \psi_2(8) = 0.0001_2 \tag{2}$$

따라서, 10진수로 나타낸 기수가 2인 Van der Corput점들은 다음과 같다.

$$\left\{\frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{8}, \frac{5}{8}, \frac{3}{8}, \frac{7}{8}, \frac{1}{16}\right\} \tag{3}$$

벡터 vdc는 이 Van der Corput열을 나타낸다. 이 결과를 출력한 그래프가 그림 4.3.3에 수록되어 있다. ■

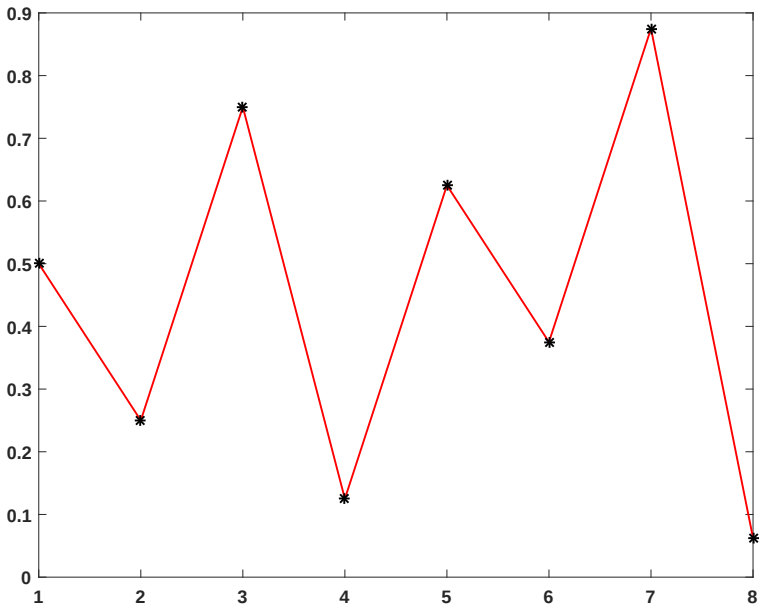


그림 4.3.3. Van der Corput열 2

Van der Corput열을 발생하는 목적은 정적분을 계산하는 것이다. 다음 예제를 살펴보자.

예제 4.3.5 Van der Corput열을 생성하고 이를 이용해서 정적분의 근사값을 구하기 위해서, 다음 MATLAB프로그램 VanDerCorputIntegral101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: VanDerCorputIntegral101.m
%  Van der Corput Integration
%  Using M-file vdcorput.m by Dimitri Shvorob
%  http://www.mathworks.com/matlabcentral/fileexchange
%    /15354-generate-a-van-der-corput-sequence/content/vdcorput.m
```

```
%-------------------------------------------------------------------------
function VanDerCorputIntegral101
TrueIntegral = int(sym('exp(x)'),0,1)
TrueInt = double(TrueIntegral)
% Van der Corput Integral
b = 2                          % Base
f = @(x) exp(x)
for jj = 1:20
    n(jj) = 1000*jj;
    VDC = vdcorput(n(jj),b);
    VDCintegral(jj) = 1/n(jj)*sum(f(VDC));
end
% Plotting
plot(n,VDCintegral,'k-o',[0 n(end)],[TrueInt TrueInt],'r-', ...
     'linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
legend('\bf Van der Corput Integral','\bf True Integral', ...
       'location','SE')
saveas(gcf,'VanDerCorputIntegral101','epsc')
save('VanDerCorputIntegral101','VDCintegral')
end
% End of program
% -------------------------------------------------------------------------
function [s] = vdcorput(k,b)
% VDCORPUT   Base-b Van der Corput sequence, elements 0,..,k
% INPUTS   : k - maximum sequence index, non-negative integer
%            b - sequence base, integer exceeding 1
% OUTPUTS  : s - (k+1)*1 array, with s(i) storing element (i+1)
%                of base-b Van der Corput sequence
% EXAMPLE  : vdcorput(0,2) = 0
%            vdcorput(1,2) = [0 0.5]'
%            vdcorput(2,2) = [0 0.5 0.25]'
% AUTHOR   : Dimitri Shvorob, dimitri.shvorob@vanderbilt.edu
if k ~= floor(k)||(k < 0)
   error('Input argument "k" must be a non-negative integer')
end
if b ~= floor(b)||(b < 2)
   error('Input argument "b" must be a positive int greater than 1')
end
s = zeros(k+1,1);
for i = 1:k
    a = basexpflip(i,b);
    g = b.^(1:length(a));
    s(i+1) = sum(a./g);
end
end
% -------------------------------------------------------------------------
function[a] = basexpflip(k,b) % reversed base-b expansion of k
j = fix(log(k)/log(b)) + 1;
a = zeros(1,j);
q = b^(j-1);
for i = 1:j
   a(i) = floor(k/q);
   k = k - q*a(i);
   q = q/b;
end
a = fliplr(a);
end
% -------------------------------------------------------------------------
```

이 MATLAB프로그램은 1000 개에서 시작해서 20000 개까지 1000 개 씩 늘려가면서 Van der Corput 열을 생성하고, 이를 사용해서 다음 정적분을 구한다.

$$\int_0^1 \exp(x)dx = \exp(1) - 1 \tag{1}$$

이 MATLAB프로그램 VanDerCorputIntegral101.m을 실행하기 위해서는 M파일들 vdcorput.m과 basexpflip.m을 필요로 한다. 이 M파일들은 D. Shvorob에 의해서 제공된 것이다.

이 MATLAB프로그램 VanDerCorputIntegral101.m을 실행하면, 그림 4.3.4 가 그려진다. 그림 4.3.4 에서 알 수 있듯이, Van der Corput 열을 이용한 수치적분값은 이론값 1.7183 에 빠르게 수렴하지는 않는다. ■

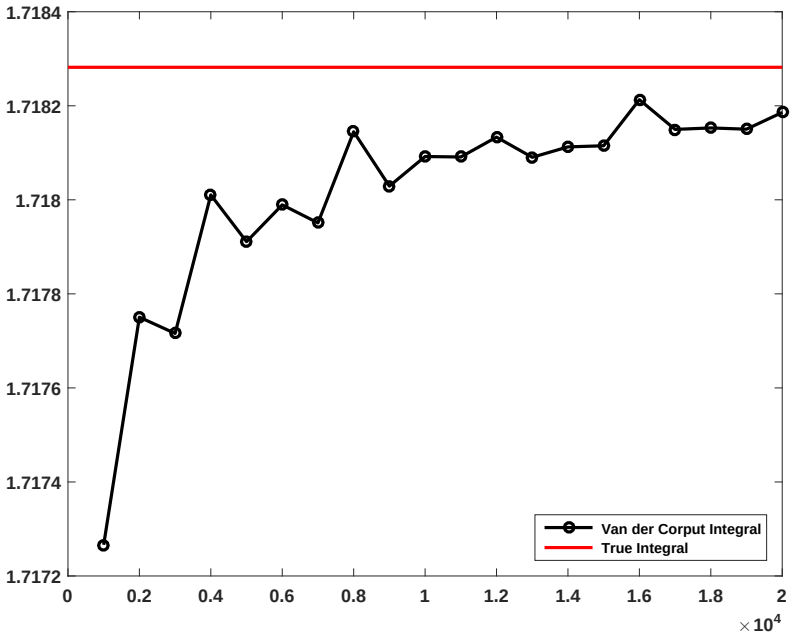


그림 4.3.4. Van der Corput 열

예제 4.3.6 몬테카를로법으로 계산한 정적분과 준몬테카를로법으로 계산한 정적분을 비교하기 위해서, 다음 MATLAB프로그램 VanDerCorputIntegral102.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: VanDerCorputIntegral102.m
%  Van Der Corput Integration 2
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all, format long
rand('twister',5489)
f = @(x) 4*sqrt(1-x.^2)
```

```
Mmax = 20;
delta = 1000;
psuedo = rand(Mmax*delta,1);
vdc = VanDerCorput101(Mmax*delta,2);
for m=1:Mmax
    Pran = f(psuedo(1:m*delta));
    MC(m) = mean(Pran);
    Qran = f(vdc(1:m*delta));
    QMC(m) = mean(Qran);
end
xx = 1:Mmax;
plot([0 Mmax],[pi pi],'b-',xx,MC,'k-*',xx,QMC,'r--o','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xtick',0:5:20)
set(gca,'xticklabel',(0:5:20)*delta)
legend('\pi','MC','QMC','location','NE')
saveas(gcf,'VanDerCorputIntegral102','epsc')
save('VanDerCorputIntegral102','MC','QMC')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 그림 4.3.5이 그려진다. 그림 4.3.5에서 흑색 실선은 몬테카를로적분을, 그리고 적색 긴점선은 준몬테카를로적분을 나타낸다. 이 그림에서 알 수 있듯이, 준몬테카를로적분이 몬테카를로적분보다 적분값 π에 빨리 수렴한다. ■

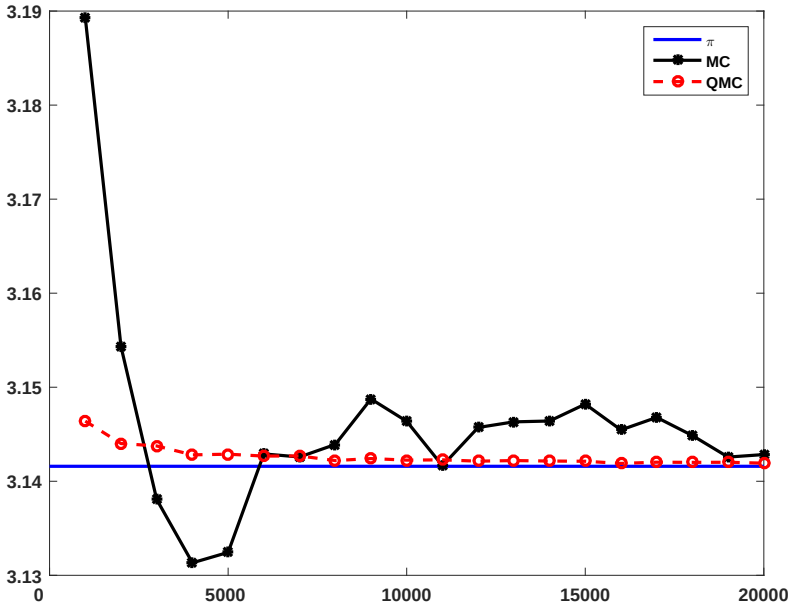


그림 4.3.5. Monte Carlo적분과 Van der Corput적분

제4.4절 Halton점열

Halton열은 생성하기가 가장 쉬운 다차원 저불일치점열이다. 서로소인 d개 기수들 $b_1, b_2, \cdots, b_d$에 대해서, 다음과 같이 정의되는 점열 $\{\boldsymbol{x}_n\}$을 Halton열이라 한다.

$$\boldsymbol{x}_n \doteq [\psi_{b_1}(n), \psi_{b_2}(n), \cdots, \psi_{b_d}(n)] \tag{4.4.1}$$

여기서 $\psi_{b_j}(\cdot)$는 기수가 b_j인 (radical-inverse function)이다. 즉, Halton열은 d개 서로수들을 기수들로 하는 Van der Corput열들을 묶은 것이다. 일반적으로 기수들 $b_1, b_2, \cdots, b_d$로 가장 작은 d개 소수들을 사용한다. 예를 들어, $d = 3$이면, $b_1 = 2$, $b_2 = 3$, 그리고 $b_3 = 5$를 사용한다.

예제 4.4.1 Halton점열을 생성하기 위해서, 다음 MATLAB프로그램 HaltonSequence101.m을 사용하자.

```
% ------------------------------------------------------------------
%  Filename: HaltonSequence101.m
%  Generating Halton Sequence
%  Programmed by CBS
%------------------------------------------------------------------
function x = HaltonSequence101(n,d)
format rat
diary HaltonSequence101.txt
% Inputs: n = No of Halton numbers, d = No of Base numbers
% Find the first d prime numbers
pri = zeros(1,d);
count = 1;
b = 1;
while count <= d
    if isprime(b) == 1
        pri(count) = b;
        count = count+1;
    end
    b = b+1;
end
pri
% (Another Method) pridum = primes(10*d); pri = pridum(1:d)
x = [(1:n)' ] ;
for jj = 1:d
    xadd = VanDerCorput101(n,pri(jj));
    x = [ x xadd' ];
end
diary off
end
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 이용해서 기수들이 3개인 Halton점열을 생성하기 위해서 MATLAB커맨드행에서 다음 명령문을 실행하라.

```
>> hal = HaltonSequence101(10,3)
```

이 MATLAB명령문이 실행되면, 다음과 같은 결과를 얻는다.

```
b =
     2     3     5


hal =
     1     1/2   1/3   1/5
     2     1/4   2/3   2/5
     3     3/4   1/9   3/5
     4     1/8   4/9   4/5
     5     5/8   7/9   1/25
     6     3/8   2/9   6/25
     7     7/8   5/9   11/25
     8     1/16  8/9   16/25
     9     9/16  1/27  21/25
    10     5/16  10/27 2/25
```

이 결과물에서 알 수 있듯이, 기수들로는 $2, 3$, 그리고 5가 사용되었고, 또한 10×3 Halton 점열이 생성되었다. ■

예제 4.4.2 Halton점열을 생성하기 위해서, 다음 MATLAB프로그램 HaltonSequenceUseHaltonset101.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: HaltonSequenceUseHaltonset101.m
%  Generating Halton Sequence Using haltonset.m
%  Programmed by CBS
%------------------------------------------------------------------
% Construct a d-dimensional point set x of Halton sequence
format rat
diary HaltonSequenceUseHaltonset101.txt
x = haltonset(3)
% Values of the point set are not generated and stored in memory
%        until you access x using net or parenthesis indexing
format rat
x5  = net(x,5)     % net.m: An intrinsic function of MATLAB
x20 = x(2:3:20,:)
diary off
% End of program
% ------------------------------------------------------------------
```

MATLAB함수 haltonset.m을 사용해서 Halton점열을 생성할 수 있다. 이 MATLAB함수 haltonset.m은 기수들이 3개인, 즉 차수가 3인 Halton점열을 행렬 x에 만들 것을 지정하는 것이다. 그러나, 이 함수를 실행했다고 해서 Halton점열이 만들어지는 것은 아니다. Halton 점열을 생성하기 위해서는 MATLAB함수 net.m을 사용하거나, 행렬 x에서 출력하고자 하는 원소들의 첨자들을 지정한다. 예를 들어, `'x(2:3:30,:)'`은 두 번째 점부터 시작해서 제 30번째 점까지 3점마다 한 점씩을 출력한다. 유의할 점은 첫 번째 Halton열은 영벡터라는 것이다.

이 MATLAB명령문이 실행되면, 다음과 같은 결과를 얻는다.

```
x5 =
         0         0         0
       1/2       1/3       1/5
       1/4       2/3       2/5
       3/4       1/9       3/5
       1/8       4/9       4/5


x20 =
       1/2       1/3       1/5
       1/8       4/9       4/5
       7/8       5/9      11/25
      5/16     10/27       2/25
     11/16     13/27      17/25
      1/32     16/27       8/25
     25/32     11/27      23/25
```

■

예제 4.4.3 Halton점열이 입방체 $[0,1]^3$에 균등하게 분포되는지를 살펴보기 위해서 다음 MATLAB프로그램 HaltonSequence102.m을 실행하자.

```
% ------------------------------------------------------------------
%  Filename: HaltonSequence102.m
%  Generating and Plotting Halton Sequence
%  Programmed by CBS
%------------------------------------------------------------------
clear all, close all, format rat
hal = HaltonSequence101(500,3);
plot3(hal(:,2),hal(:,3),hal(:,4),'k.','linewidth',1)
grid on
set(gca,'fontsize',11,'fontweigh','bold')
```

```
saveas(gcf,'HaltonSequence102a','epsc')
figure
subplot(2,2,1)
plot(hal(:,2),hal(:,3),'k.','linewidth',1)
set(gca,'fontsize',11,'fontweigh','bold')
axis square
subplot(2,2,2)
plot(hal(:,2),hal(:,4),'k.','linewidth',1)
set(gca,'fontsize',11,'fontweigh','bold')
axis square
subplot(2,2,3)
plot(hal(:,3),hal(:,4),'k.','linewidth',1)
set(gca,'fontsize',11,'fontweigh','bold')
axis square
subplot(2,2,4)
psuedo = rand(500,3);
plot(psuedo(:,1),psuedo(:,2),'k.','linewidth',1)
set(gca,'fontsize',11,'fontweigh','bold')
axis square
saveas(gcf,'HaltonSequence102b','epsc')
save('HaltonSequence102','hal')
% End of program
% --------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 그림 4.4.1과 그림 4.4.2가 그려진다. 그림 4.4.1은 500×3 Halton점열 $\{\boldsymbol{x}_n \doteq [\Phi_2(n), \Phi_3(n), \Phi_5(n)]\}$을 그린 것이다. 그림 4.4.2의 좌측상단 그래프에는 $\{[\Phi_2(n), \Phi_3(n)]\}$을, 우측상단 그래프에는 $\{[\Phi_2(n), \Phi_5(n)]\}$을, 좌측하단 그래프에는 $\{[\Phi_3(n), \Phi_5(n)]\}$을, 그리고 우측하단 그래프에는 500×2 의사일양난수열을 그린 것이다. 이 그래프들에서 알 수 있듯이, 의사일양난수들보다는 Halton점열들이 더 균등하게 분포되어 있다. ■

Halton점열의 스타디스크레펀시는 다음 식을 만족한다.

$$D_n^*(S) = O\left(\frac{[\log n]^d}{n}\right) \tag{4.4.2}$$

정의 4.2.4와 식 (4.4.2)에서 알 수 있듯이, Halton점열은 저불일치점열이다. 또한, 낮은 차수 d의 초입방체에서 Halton점열이 균등하게 분포한다고 할 수 있다. Niederreiter [37]는 식 (4.4.2)의 비례상수가 점근적으로 d^d에 수렴함을 보였다. 따라서, 차수 d가 높은 경우에는 초입방체에서 Halton점열이 균등하게 분포하지 않는 경우가 나타난다.

예제 4.4.4 차원이 높은 경우 Halton점열이 균등하게 분포하지 않음을 보이기 위해서, 다음 MATLAB프로그램 HaltonSequenceHighDim101.m을 실행해 보자.

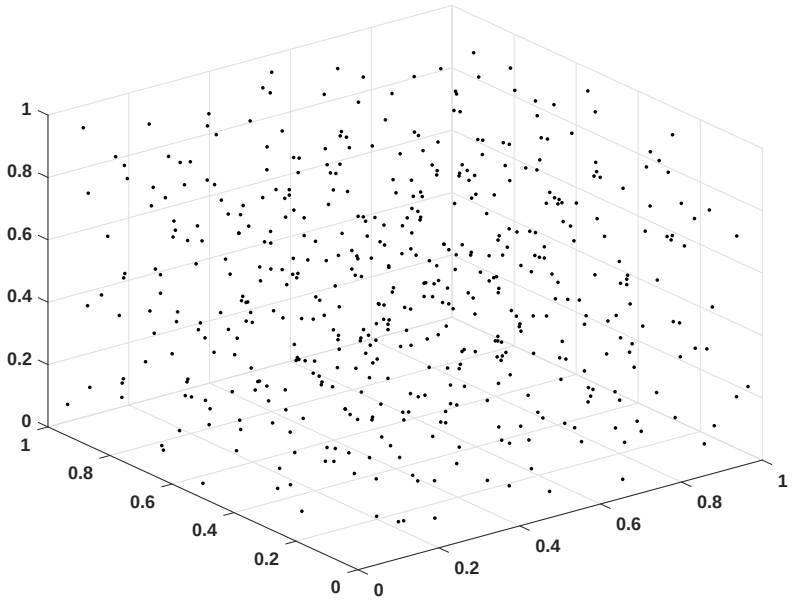


그림 4.4.1. Halton점열

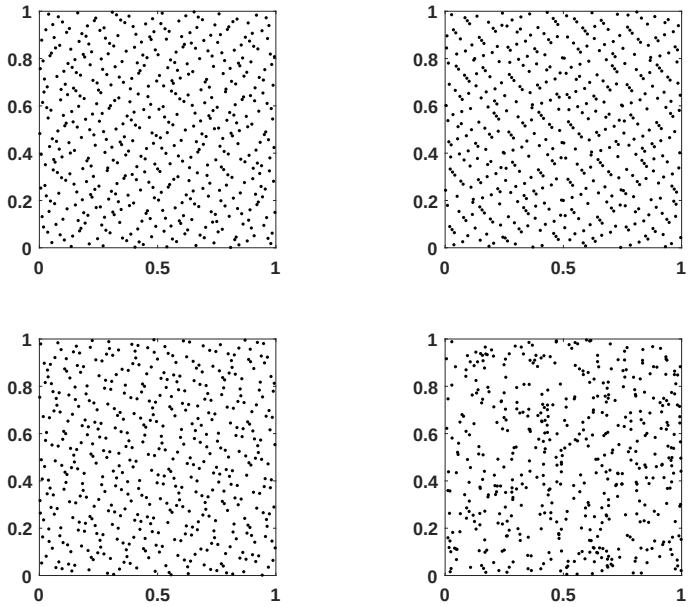


그림 4.4.2. Halton점열과 의사일양난수열

```
% ------------------------------------------------------------------------
%  Filename: HaltonSequenceHighDim101.m
%  High Dimenional Faure Sequence
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all
n = 1000, d = 19
hal = HaltonSequence101(n,d);
plot(hal(:,d-1),hal(:,d),'k.','linewidth',1)
```

```
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf x_{18}')
ylabel('\bf x_{19}','rotation',0)
axis square
saveas(gcf,'HaltonSequenceHighDim101','epsc')
save('HaltonSequenceHighDim101','hal')
% End of program
% --------------------------------------------------------------------
```

이 MATLAB명령문이 실행되면, 차원 d가 19인 Halton점열을 생성한다. 이 Halton점열의 제18번째 원소 대 제19번째 원소의 산점도가 그림 4.4.3에 그려져 있다. 이 그림에서 알 수 있듯이, 높은 차원의 Halton점열은 초입방체에서 균등하게 분포한다고 할 수 없다. ■

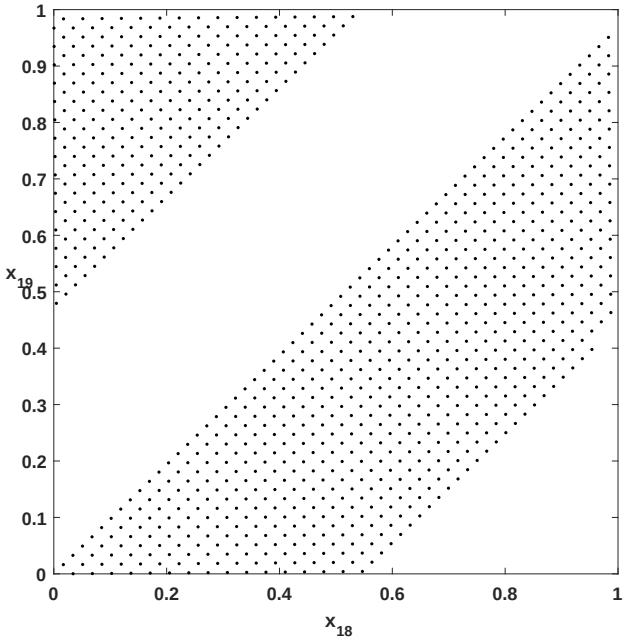


그림 4.4.3. 고차원 Halton점열

다음 예제에서 알 수 있듯이, 기수를 잘못 고르면 생성된 Halton점열이 균등하게 분포하지 않는다.

예제 4.4.5 잘못 선택된 기수가 좋지 않은 Halton점열을 생성하는 예를 보기 위해서, 다음 MATLAB프로그램 HaltonBadBase101.m을 실행해 보자.

```
% --------------------------------------------------------------------
%  Filename: HaltonBadBase101.m
%  Bad Base for Halton Sequence
%  Programmed by CBS
%--------------------------------------------------------------------
clear all, close all
n = 50
```

```
rand('twister',5489)
psuedo = rand(50,2);
vdc2 = VanDerCorput101(50,2);
vdc3 = VanDerCorput101(50,3);
vdc4 = VanDerCorput101(50,4);
vdc43 = VanDerCorput101(50,43);
vdc47 = VanDerCorput101(50,47);
subplot(2,2,1)
plot(psuedo(:,1),psuedo(:,2),'k.','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf Psuedo-Random'),ylabel('\bf Psuedo-Random'), grid on
subplot(2,2,2)
plot(vdc2,vdc3,'k.','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf Base 2'),ylabel('\bf Base 3'), grid on
subplot(2,2,3)
plot(vdc2,vdc4,'k.','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf Base 2'),ylabel('\bf Base 4'), grid on
subplot(2,2,4)
plot(vdc43,vdc47,'k.','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf Base 43'),ylabel('\bf Base 47'), grid on
saveas(gcf,'HaltonBadBase101','epsc')
save('HaltonBadBase101','vdc43','vdc47')
% End of program
% --------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 그림 4.4.4가 그려진다. 그림 4.4.4의 좌측상단 그래프는 일양난수들의 산점도이다. 우측상단의 그래프는 기수가 2인 Van der Corput열과 기수가 3인 Van der Corput열의 산점도이다. 좌측하단의 그래프는 기수가 2인 Van der Corput열과 기수가 4인 Van der Corput열이다. 이 그래프에서 알 수 있듯이, 기수가 소수가 아니면 비록 작다하더라도 균등하게 분포된 열을 발생시키지 못한다. 우측하단의 그래프는 기수가 43인 Van der Corput열과 기수가 47인 Van der Corput열이다. 이 그래프에서 알 수 있듯이, 기수가 소수라 하더라도 크면, 균등하게 분포된 열을 발생시키지 못한다. 따라서, Van der Corput열과 Halton점열에서는 작은 소수들을 기수로 사용해야 한다. 또한, 기수가 소수가 아니거나 큰 경우에는 준난수보다 의사난수가 더 균등하게 분포함을 추측할 수 있다. ■

예제 4.4.6 몬테카를로법으로 계산한 정적분과 준몬테카를로법으로 계산한 정적분을 비교하기 위해서, 다음 MATLAB프로그램 HaltonIntegral101.m을 실행해 보자.

```
% --------------------------------------------------------------------------
%  Filename: HaltonIntegral101.m
%  Halton 2-D Integration 1
%  Programmed by CBS
%--------------------------------------------------------------------------
clear all, close all, format long
```

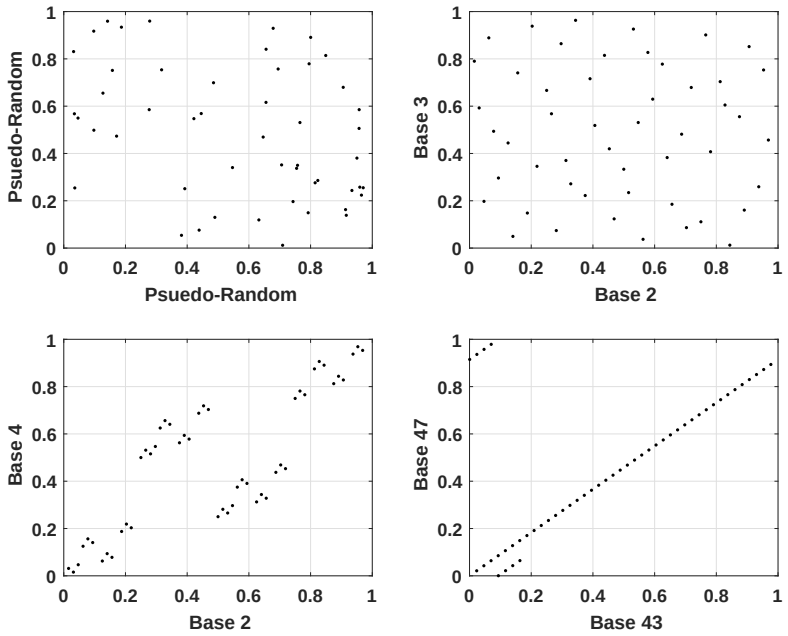


그림 4.4.4. 기수와 Van der Corput 열

```
f = @(x,y) exp(-x.*y).*(sin(pi*x)+cos(pi*y));
Tint = dblquad(f,0,1,0,1)
Mmax = 20;
delta = 100;
rand('twister',5489)
psuedo = rand(Mmax*delta,2);
vdc2 = VanDerCorput101(Mmax*delta,2);
vdc3 = VanDerCorput101(Mmax*delta,3);
for m=1:Mmax
    Pran = f(psuedo(1:m*delta,1),psuedo(1:m*delta,2));
    MC(m) = mean(Pran);
    Qran = f(vdc2(1:m*delta),vdc3(1:m*delta));
    QMC(m) = mean(Qran);
end
% Plotting
xx = 1:Mmax;
plot([0 Mmax],[Tint Tint],'b-',xx,MC,'k-*', ...
    xx,QMC,'r--o','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xtick',0:5:20)
set(gca,'xticklabel',(0:5:20)*delta)
legend('Value','MC','Halton-QMC','location','NE')
saveas(gcf,'HaltonIntegral101','epsc')
save('HaltonIntegral101','MC','QMC')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램은 다음 정적분을 몬테카를로법과 Halton점열을 사용한 준몬테카를로법으로 계산하기 위한 것이다.

$$I \doteq \int_0^1 \int_0^1 \exp(-xy)\left[\sin \pi x + \cos \pi y\right] dxdy \tag{1}$$

구적법(quadrature)을 실행하는 MATLAB함수 dblquad.m을 사용하면, 다음 식이 성립함을 알 수 있다.

$$I = 0.578396241990147 \tag{4.4.3}$$

이 MATLAB을 실행하면, 그림 4.4.5가 그려진다. 그림 4.4.5에서 흑색 실선은 몬테카를로적분을, 그리고 적색 긴점선은 Halton점열을 사용한 준몬테카를로적분을 나타낸다. 이 그림에서 알 수 있듯이, 준몬테카를로법에 의한 수치적분값이 몬테카를로법에 의한 수치적분값보다 더 빨리 적분값에 수렴한다. ■

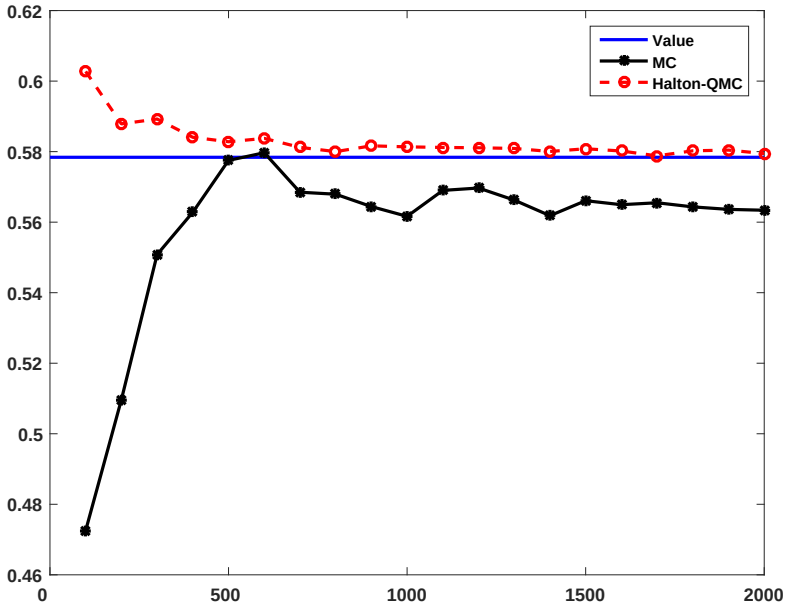


그림 4.4.5. Monte Carlo적분과 Halton적분

다음 예제는 몬테카를로법과 준몬테카를로법을 사용해서 유럽형옵션의 가치를 평가하는 것이다.

예제 4.4.7 몬테카를로법으로 평가한 유럽형옵션의 가치와 준몬테카를로법으로 계산한 가치를 비교하기 위해서, 다음 MATLAB프로그램 QMCeuropeanCallPut101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: QMCeuropeanCallPut101.m
%  European Call and Put Option Pricing
%           using Quasi-Monte Carlo Method 1
%  Programmed by CBS
%------------------------------------------------------------------------
function QMCeuropeanCallPut101
```

```
clear all, close all, format long
rd = 0.05       % rd = domestic interest rate;
rf = 0.02       % rf = foreign interest rate
sigma = 0.30    % sigma = volatility
St = 1100       % St = current price of underlyingt
ctime = 0       % ctime = current time t;
etime = 1/4     % etime = expiry T
K = 1050        % K = strike price
tau = etime - ctime;
sqtau = sqrt(tau);
% Genreating Normal Random Numbers using Box-Muller transform
rand('twister',5489)
Mmax = 20; delta = 10000;
for jj = 1:Mmax
    N = jj*delta;
    N2 = ceil(N/2);
    U = haltonset(1);
    halt = U(1:2*N2,1);
    Qrand = norminv(halt);
    Prand = randn(2*N2,1);
% Generating Random Numbers from S(T)
    STQ = St*exp( (rd-rf-0.5*sigma^2)*tau + sigma*sqtau*Qrand );
    STP = St*exp( (rd-rf-0.5*sigma^2)*tau + sigma*sqtau*Prand );
% European Call Price by QMC method and MC method
    CQMC(jj) = exp( -rd*tau )*mean(max(STQ-K,0));
    CMC(jj) = exp( -rd*tau )*mean(max(STP-K,0));
% European Put Price by QMC method and MC method
    PQMC(jj) = exp( -rd*tau )*mean(max(K-STQ,0));
    PMC(jj) = exp( -rd*tau )*mean(max(K-STP,0));
end
% Black-Scholes Solution
[bsCt , bsPt] = BSeuCP1(rd,rf,sigma,St,ctime,etime,K);
% Plotting
xx = 1:Mmax;
plot([0 Mmax],[bsCt bsCt],'b-',xx,CMC,'k-*', ...
      xx,CQMC,'r--o','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xtick',0:5:20)
set(gca,'xticklabel',(0:5:20)*delta)
legend('Black-Scholes Price','MC price','QMC price')
saveas(gcf,'QMCeuropeanCallPut101','epsc')
save('QMCeuropeanCallPut101','CQMC','PQMC')
end
% End of program
% ------------------------------------------------------------------------
%  Filename: BSeuCP1.m
%  Black-Scholes European Call & Put Option Pricing
%  Programmed by CBS
%------------------------------------------------------------------------
function [Ct , Pt] = BSeuCP1(rd,rf,sigma,St,ctime,etime,K)
tau = etime - ctime;
d1 = 1/(sigma*sqrt(tau))*(log(St/K)+(rd-rf+sigma*sigma/2)*tau);
d2 = d1 - sigma*sqrt(tau);
Ct = St*exp(-rf*tau)*normcdf(d1) - K*exp(-rd*tau)*normcdf(d2);
Pt = K*exp(-rd*tau)*normcdf(-d2) - St*exp(-rf*tau)*normcdf(-d1);
% The current option values are
disp('Call option value ='), disp(Ct)
disp('Put option value ='),  disp(Pt)
end
% ------------------------------------------------------------------------
```

이 MATLAB 프로그램은 한국의 무위험이자율이 $r_d = 0.05$, 미국의 무위험이자율이 $r_f = 0.02$, 원자산인 USD/KRW의 변동성이 $\sigma = 0.30$, 현재시점 $t = 0$에서 환율이 $S_t = 1100$인 경우에, 만기시점 $T = 0.25$에서 행사가격이 $K = 1050$인 유럽형콜옵션의 가치와 유럽형 풋옵션의 가치를 평가하기 위한 것이다.

MATLAB함수 haltonset.m을 사용해서 발생시킨 준난수에 역함수법을 적용해서 생성한 정규준난수(normal quasi-random number)가 행렬 Qrand에 저장되고, MATLAB함수 randn.m을 사용해서 발생시킨 정규난수는 행렬 Prand에 저장한다. 이들을 사용해서 계산한 유럽형콜옵션가치는 각각 CQMC와 CMC이고, 유럽형풋옵션가치는 각각 PQMC와 PMC이다. 또한, M-파일 BSeuCP1.m을 사용해서 계산한 유럽형콜옵션과 유럽형풋옵션의 Black-Scholes가치들은 bsCt와 bsPt이다.

이 MATLAB프로그램을 실행하면, 유럽형콜옵션과 유럽형풋옵션의 Black-Scholes가치들은 각각 96.5726과 39.0156임을 알 수 있고, 그림 4.4.6이 그려진다. 그림 4.4.6에서 흑색 실선은 몬테카를로법에 의한 유럽형콜옵션가치를, 적색 긴점선은 준몬테카를로법에 의한 유럽형콜옵션가치를, 그리고 청색 실선은 Black-Scholes가치를 나타낸다. 이 그림에서 알 수 있듯이, 이 옵션가치평가문제에서는 준몬테카를로적분이 몬테카를로적분보다 더 빨리 Black-Scholes가치에 수렴한다. ■

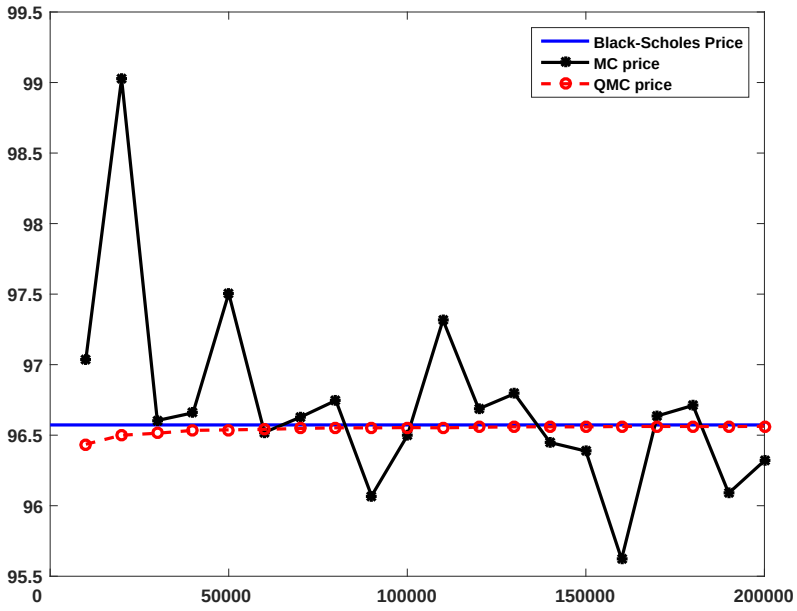


그림 4.4.6. 준몬테카를로법에 의한 유럽형콜옵션의 가치

몬테카를로법과 달리 준몬테카를로법은 대수법칙을 이용하는 것이 아니다. 따라서, 확

률개념이 들어간 성질을 바탕으로 하는 기법을 준몬테카를로법과 같이 사용할 때는 주의를 기울여야 한다.

예제 4.4.8 Box-Müller변환은 결합일양확률벡터에서 결합정규확률벡터를 유도하는 이다. 따라서, 준몬테카를로법을 적용할 때 이 Box-Müller변환을 함께 사용하는데는 주의를 기울여야 한다. 다음 MATLAB프로그램 QMCeuropeanCallPutBoxMuller101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: QMCeuropeanCallPutBoxMuller101.m
%  European Call and Put Option Pricing using Quasi-Monte Carlo Method
%           and Box-Muller Transform
%  Programmed by CBS
%------------------------------------------------------------------------
function QMCeuropeanCallPutBoxMuller101

clear all, close all, format long
rd = 0.05       % rd = domestic interest rate;
rf = 0.02       % rf = foreign interest rate
sigma = 0.30    % sigma = volatility
St = 1100       % St = current price of underlyingt
ctime = 0       % ctime = current time t;
etime = 1/4     % etime = expiry T
K = 1050        % K = strike price
tau = etime - ctime;
sqtau = sqrt(tau);
Mmax = 20; delta = 1000; Mdelta = Mmax*delta;
xx = (1:Mmax)*delta;
N = Mmax*delta;
N2 = ceil(N/2);
Prand = randn(2*N2,1);
% Black-Scholes Solution
[bsCt , bsPt] = BSeuCP1(rd,rf,sigma,St,ctime,etime,K);
% Genreating Normal Random Numbers using bases = 2, 3
U = haltonset(2);
u1 = U(N2,1); u2 = U(N2,2);
SqLog = sqrt(-2*log(u1));
BMrand(1:N2,1) = SqLog.*cos(2*pi*u2);
BMrand(N2+1:2*N2,1) = SqLog.*sin(2*pi*u2);
[C1 P1]= QMC_MCeuropeanCallPut(rd,rf,sigma,St,ctime,etime,K,BMrand)
% Plotting
subplot(2,2,1)
plot([0 Mdelta],[bsCt bsCt],'b-',xx,C1,'r-*','linewidth',1.5)
set(gca,'fontsize',11,'fontweigh','bold')
% Genreating Normal Random Numbers using bases = 2, 7
u1 = VanDerCorput1(N2,2);
u2 = VanDerCorput1(N2,7);
SqLog = sqrt(-2*log(u1));
BMrand(1:N2,1) = SqLog.*cos(2*pi*u2);
BMrand(N2+1:2*N2,1) = SqLog.*sin(2*pi*u2);
[C2 P2]= QMC_MCeuropeanCallPut(rd,rf,sigma,St,ctime,etime,K,BMrand);
   % Plotting
subplot(2,2,2)
plot([0 Mdelta],[bsCt bsCt],'b-',xx,C2,'r-*','linewidth',1.5)
set(gca,'fontsize',11,'fontweigh','bold')
% Genreating Normal Random Numbers using bases = 2, 11
u1 = VanDerCorput1(N2,2);
u2 = VanDerCorput1(N2,11);
```

```
SqLog = sqrt(-2*log(u1));
BMrand(1:N2,1) = SqLog.*cos(2*pi*u2);
BMrand(N2+1:2*N2,1) = SqLog.*sin(2*pi*u2);
[C3 P3]= QMC_MCeuropeanCallPut(rd,rf,sigma,St,ctime,etime,K,BMrand);
   % Plotting
subplot(2,2,3)
plot([0 Mdelta],[bsCt bsCt],'b-',xx,C3,'r-*','linewidth',1.5)
set(gca,'fontsize',11,'fontweigh','bold')
% Genreating Normal Random Numbers using bases = 7, 11
u1 = VanDerCorput1(N2,7);
u2 = VanDerCorput1(N2,11);
SqLog = sqrt(-2*log(u1));
BMrand(1:N2,1) = SqLog.*cos(2*pi*u2);
BMrand(N2+1:2*N2,1) = SqLog.*sin(2*pi*u2);
[C4 P4]= QMC_MCeuropeanCallPut(rd,rf,sigma,St,ctime,etime,K,BMrand);
   % Plotting
subplot(2,2,4)
plot([0 Mdelta],[bsCt bsCt],'b-',xx,C4,'r-*','linewidth',1.5)
set(gca,'fontsize',11,'fontweigh','bold')
saveas(gcf,'QMCeuropeanCallPutBoxMuller101','epsc')
end
% End of program
% ------------------------------------------------------------------------
%  Filename: BSeuCP1.m
%  Black-Scholes European Call & Put Option Pricing
%------------------------------------------------------------------------
function [Ct , Pt] = BSeuCP1(rd,rf,sigma,St,ctime,etime,K)

% rd = domestic risk-free interest rate
% rf = foreign risk-free interest rate
% sigma = volatility
% St = current price of underlying asset S_t
% ctime = current time t
% etime = expiry T
% K = strike price
% Ct = European Call Option Price by Black and Scholes
% Pt = European Put Option Price by Black and Scholes
%
%
tau = etime - ctime;
d1 = 1/(sigma*sqrt(tau))*( log(St/K) + (rd - rf + sigma*sigma/2)*tau );
d2 = d1 - sigma*sqrt(tau);
Ct = St*exp(-rf*tau)*normcdf(d1) - K*exp(-rd*tau)*normcdf(d2);
Pt = K*exp(-rd*tau)*normcdf(-d2) - St*exp(-rf*tau)*normcdf(-d1);
% The current option values are
disp('Call option value ='), disp(Ct)
disp('Put option value ='), disp(Pt)

end
% end of program
% ------------------------------------------------------------------------
%  Filename: QMC_MCeuropeanCallPut.m
%  Subroutine for European Call and Put Option Pricing
%       using Quasi-Monte Carlo and Monte Carlo Methods
%------------------------------------------------------------------------
function [Cp Pp] = QMC_MCeuropeanCallPut(rd,rf,sigma,St,ctime,etime,K,RN)

tau = etime - ctime;
sqtau = sqrt(tau);
% Genreating Normal Random Numbers using Box-Muller transform
Mmax = 20; delta = 1000;
```

```
for jj = 1:Mmax
    Qrand = RN(1:jj*delta,1);
    STT = St*exp( (rd-rf-0.5*sigma^2)*tau + sigma*sqtau*Qrand);
% European Call Price by QMC method and MC method
    Cp(jj) = exp( -rd*tau )*mean(max(STT-K,0));
% European Put Price by QMC method and MC method
    Pp(jj) = exp( -rd*tau )*mean(max(K-STT,0));
end

end
% End of program
% ------------------------------------------------------------------------
%  Filename: VanDerCorput1.m
%  Generating van der Corput Sequence
%------------------------------------------------------------------------
function vdc = VanDerCorput1(n,b)

% Input: n = No of van der Corput numbers
r = floor(log(n)/log(b)+1);
nn = 1:n;
Amat = [ ];
for jj=0:r-1
    a = mod(nn,b);
    nn = (nn-a)/b;
    Amat = [ Amat; a ];
end
vdc = b.^(-(1:r))*Amat;

end
% (Another Method) x = haltonset(1); vdc = net(x,n+1)       % for base = 2
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램은 한국의 무위험이자율이 $r_d = 0.05$, 미국의 무위험이자율이 $r_f = 0.02$, 원자산인 USD/KRW의 변동성이 $\sigma = 0.30$, 현재시점 $t = 0$에서 환율이 $S_t = 1100$인 경우에, 만기시점 $T = 0.25$에서 행사가격이 $K = 1050$인 유럽형콜옵션과 유럽형풋옵션의 가치들을 평가하기 위한 것이다. 우선, 다양한 기수들의 쌍에 대한 2차원 Halton점열들을 생성하고, 이 준난수들에 Box-Müller변환을 적용해서 정규준난수들을 생성한다. 이렇게 생성된 정규준난수들을 1000개부터 1000개씩 20000개까지 증가시켜가며 유럽형콜옵션가치와 유럽형풋옵션가치를 계산한다.

이 MATLAB프로그램을 실행하면, 유럽형콜옵션과 유럽형풋옵션의 Black-Scholes가치들은 각각 96.5726과 39.0156임을 알 수 있고, 또한 그림 4.4.7이 그려진다. 첫째, MATLAB함수 haltonset.m을 사용해서 발생시킨 준난수에 Box-Müller변환을 적용해서 정규준난수들을 생성한다. MATLAB명령문 'U = haltonset(2)' 에서는 기수들로 2와 3을 사용함을 상기하라. 이 정규준난수들을 사용해서 유럽형콜옵션가치 C1과 유럽형풋옵션가치 P1을 계산한다. 이렇게 계산된 유럽형콜옵션가치를 그린 것이 그림 4.4.7의 좌측상단 그래프이다. 이 그래프에서 알 수 있듯이, 기수들이 2와 3인 준난수열들과 Box-Müller변환을 사용해서 발생한 정규준난수열은

유럽형콜옵션의 가치를 평가하는데 적절하지가 않다. 이는 준난수 자체의 문제가 아니라 Box-Müller변환에 기인하는 것이라고 생각된다. 기수 2에 해당하는 준난수열 $\{u_{1,k}\}$와 기수 3에 해당하는 준난수열 $\{u_{2,k}\}$에서 $u_{1,k}$와 $u_{2,k}$는 자연수 k의 근기역함수값들 $\psi_2(k)$와 $\psi_3(k)$이다. 이 경우에는 기수들이 비슷하므로, 준난수열들 $\{u_{1,k}\}$와 $\{u_{2,k}\}$이 상관관계를 갖는다. 이는 Box-Müller변환이 필요로하는 가정에 어긋난다. 둘째, M-파일 VanDerCorput1.m을 사용해서 발생시킨 준난수에 Box-Müller변환을 적용해서 정규준난수들을 생성한다. MATLAB 명령문 `'u1 = VanDerCorput1(N2,2)'`에서는 기수를 2로 하는 Van der Corput열을, 그리고 MATLAB명령문 `'u2 = VanDerCorput1(N2,7)'`에서는 기수를 7로 하는 Van der Corput 열을 생성한다. 즉, 기수들이 $(2,7)$인 Halton점열을 생성한다. 이 Halton점열에 Box-Müller 변환을 적용해서 생성한 정규준난수들을 사용해서 유럽형콜옵션가치 C2와 유럽형풋옵션가치 P2를 계산한다. 이렇게 계산된 유럽형콜옵션가치를 그린 것이 그림 4.4.7의 우측상단 그래프이다. 셋째, 같은 방법으로 기수들이 $(2,11)$인 Halton점열을 생성한다. 이 Halton점열에 Box-Müller변환을 적용해서 생성한 정규준난수들을 사용해서 유럽형콜옵션가치 C3와 유럽형풋옵션가치 P3를 계산한다. 이렇게 계산된 유럽형콜옵션가치를 그린 것이 그림 4.4.7의 좌측하단 그래프이다. 넷째, 같은 방법으로 기수들이 $(7,11)$인 Halton점열을 생성한다. 이 Halton점열에 Box-Müller변환을 적용해서 생성한 정규준난수들을 사용해서 유럽형콜옵션가치 C4와 유럽형풋옵션가치 P4를 계산한다. 이렇게 계산된 유럽형콜옵션가치를 그린 것이 그림 4.4.7의 우측하단 그래프이다. 그림 4.4.7에서 알 수 있듯이, Halton점열에 Box-Müller 변환을 적용할 때 두 기수들은 너무 가까우면 안된다. 그 이유는 각 차원의 Van der Corput 열들이 독립성을 잃기 때문이다. 또한, 예제 4.4.8에서 알 수 있듯이, 너무 큰 기수를 사용하면 안 된다. ■

제4.5절 Faure점열

제4.4절에서 알 수 있듯이, 고차원 Halton점열은 높은 상관관계를 갖는 문제점을 지니고 있다. 이러한 문제점을 피해가기 위해서 Faure점열을 사용할 수 있다. 앞에서 설명했듯이, d차원 Halton점열은 d개 기수들을 사용하므로 d가 큰 경우에는 가장 큰 기수가 클 수 밖에 없다. 반면에, Faure점열은 d차원 문제에서 d 이상인 소수 하나를 기수로 사용하는 저불일치점열이다. 일반적으로 d 이상인 소수 중에서 가장 작은 것을 Faure점열의 기수 b로 사용한다.

Faure점열은 다음과 같이 구성된다. 먼저 첫 번째 차원의 점열을 구하기 위해서는 기수가

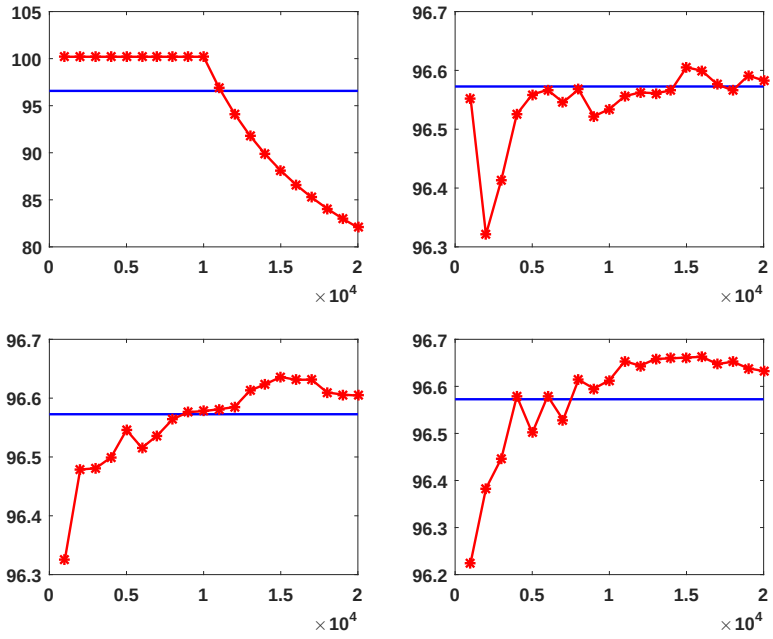


그림 4.4.7. Halton점열과 Box-Müller변환

b인 Van der Corput열 $\left\{\psi_b(n)=\sum_{l=0}^{\infty}a_l(n)b^{-l-1}\right\}$을 이용한다. 여기서 $\{a_l(n)\}$은 다음 식을 만족한다.

$$n=\sum_{l=0}^{\infty}a_l(n)b^l,\quad (n=0,1,\cdots) \tag{4.5.1}$$

각 $i(=1,2,\cdots,d)$와 각 $j(=0,1,\cdots)$에 대해서 다음 상수 $y_j^{(i)}(n)$을 정의하자.

$$y_j^{(i)}(n)\doteq\sum_{l=j}^{\infty}\binom{l}{j}[i-1]^{l-j}a_l(n)\pmod b \tag{4.5.2}$$

제n번째 점의 제i번째 원소 $y^{(i)}(n)$을 다음과 같이 정의하자.

$$y^{(i)}(n)\doteq\sum_{j=0}^{\infty}y_j^{(i)}(n)b^{-j-1} \tag{4.5.3}$$

벡터열 $\{\boldsymbol{y}_n\doteq\left[y^{(1)}(n),y^{(2)}(n),\cdots,y^{(d)}(n)\right]\}$을 Faure점열이라 한다. 식 (4.5.2)와 식 (4.5.3)에서 알 수 있듯이, $y^{(1)}(n)=\psi_b(n)$이다.

식 (4.3.5)에서 알 수 있듯이, 각 $j(\geq r+1)$에 대해서 식 $y_j^{(i)}(n)=0$을 만족하는 자연수 r이 존재한다. 따라서, 식 (4.5.3)을 다음과 같이 쓸 수 있다.

$$y^{(i)}(n)=\sum_{j=0}^{r-1}y_j^{(i)}(n)b^{-j-1} \tag{4.5.4}$$

각 $i(=1,2,\cdots,d)$에 대해서 다음 행렬을 정의하자.

$$C^{(i)} \doteq \begin{bmatrix} \binom{0}{0}i^0 & \binom{1}{0}i^1 & \binom{2}{0}i^2 & \cdots & \binom{r-1}{0}i^{r-1} \\ 0 & \binom{1}{1}i^0 & \binom{2}{1}i^1 & \cdots & \binom{r-1}{1}i^{r-2} \\ 0 & 0 & \binom{2}{2}i^0 & \cdots & \binom{r-1}{2}i^{r-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \binom{r-1}{r-1}i^0 \end{bmatrix} \tag{4.5.5}$$

다음 식들이 성립함을 알 수 있다.

$$C^{(i)} = C^{(1)}C^{(i-1)}, \quad (i=1,2,\cdots,d) \tag{4.5.6}$$

이러한 $C^{(i)}$를 생성행렬(generator matrix)이라 부른다. 식 (4.5.2)에서 알 수 있듯이, 다음 식들이 성립한다.

$$\boldsymbol{y}_r^{(i)}(n) \doteq \begin{bmatrix} y_0^{(i)}(n) \\ y_1^{(i)}(n) \\ \vdots \\ y_{r-1}^{(i)}(n) \end{bmatrix} = C^{(i-1)} \begin{bmatrix} a_0(n) \\ a_1(n) \\ \vdots \\ a_{r-1}(n) \end{bmatrix} \pmod b, \quad (i=1,2,\cdots,d) \tag{4.5.7}$$

여기서 $C^{(0)} = I_r$이다. 다음과 같은 벡터 $\boldsymbol{\beta}_r$을 정의하자.

$$\boldsymbol{\beta}_r \doteq \left[\frac{1}{b}, \frac{1}{b^2}, \cdots, \frac{1}{b^r}\right]^t \tag{4.5.8}$$

식 (4.5.7)과 식 (4.5.8)을 식 (4.5.4)에 대입하면, 다음 식을 얻는다.

$$y^{(i)}(n) = \boldsymbol{y}_r^{(i)}(n)^t \boldsymbol{\beta}_r \tag{4.5.9}$$

집합 $\{0,1,\cdots,b^r-1\}$의 각 원소를 기수 b로 전개할 때, 각 원소를 r개 이하 항들로 나타낼 수 있다. 집합 $\{\boldsymbol{a}(n) = [a_0(n), a_1(n), \cdots, a_{r-1}(n)]^t\}$은 원소들을 $\{0,1,\cdots,b-1\}$로 하는 b^r개의 모든 벡터들을 포함한다. 따라서, 각 $n(=0,1,\cdots,b^r-1)$에 대해서 $C^{(i)}\boldsymbol{a}(n) \pmod b$는 벡터 $\boldsymbol{a}(n)$의 순열(permutation)이다. 또한, 각 $j(=0,1,\cdots,b-1)$와 각 $n\,(=jb^r, jb^r+1, \cdots, [j+1]b^r-1)$에 대해서도 같은 성질이 성립함을 알 수 있다. 따라서, Faure점열의 제n번째 점 $\boldsymbol{y}(n)$의 제i번째 좌표 $y^{(i)}(n)$의 집합

$\left\{y^{(i)}(n) \mid n = jb^r, jb^r+1, \cdots, [j+1]b^r - 1\right\}$은 Van der Corput열 $\{\psi_b(n) \mid n = jb^r, jb^r + 1, \cdots, [j+1]b^r - 1\}$의 순열이다.

예제 4.5.1 만약 $r = 2$이고 $b = 3$이면, 생성행렬들은 다음과 같다.

$$C^{(1)} = \begin{bmatrix} \binom{0}{0}1^0 & \binom{1}{0}1^1 \\ 0 & \binom{1}{1}1^0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \tag{1}$$

$$C^{(2)} = \begin{bmatrix} \binom{0}{0}2^0 & \binom{1}{0}2^1 \\ 0 & \binom{1}{1}2^0 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} \tag{2}$$

식 (1)과 식 (2)로부터 다음 식이 성립함을 확인할 수 있다.

$$C^{(2)} = C^{(1)}C^{(1)} \tag{3}$$

식 (4.5.7)에서 알 수 있듯이, $\{\boldsymbol{y}_2^{(1)}(n) = \boldsymbol{a}(n) \mid n = 0, 1, \cdots, 3^2 - 1\}$은 다음과 같다.

$$\left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \end{bmatrix} \right\} \tag{4}$$

식 (1)과 식 (4)에서 알 수 있듯이, $\{\boldsymbol{y}_2^{(2)}(n) = C^{(1)}\boldsymbol{a}(n) \pmod 3 \mid n = 0, 1, \cdots, 3^2 - 1\}$은 다음과 같다.

$$\left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \begin{bmatrix} 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix} \right\} \tag{5}$$

식 (5)의 벡터들은 식 (4)의 벡터들의 순열이다. 식 (2)와 식 (4)에서 알 수 있듯이, $\left\{\boldsymbol{y}_2^{(3)}(n) = C^{(2)}\boldsymbol{a}(n) \pmod 3 \mid n = 0, 1, \cdots, 3^2 - 1\right\}$은 다음과 같다.

$$\left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \begin{bmatrix} 0 \\ 2 \end{bmatrix} \right\} \tag{6}$$

식 (6)의 벡터들 또한 식 (4)의 벡터들의 순열이다.

식 (4.5.8)에서 알 수 있듯이, 다음 식이 성립한다.

$$\boldsymbol{\beta}_2 \doteq \left[\frac{1}{3}, \frac{1}{9}\right]^t \tag{7}$$

식 (4)와 식 (7)을 식 (4.5.9)에 대입하면, $\left\{y^{(1)}(n)=\boldsymbol{\beta}_2^t\boldsymbol{y}_2^{(1)}(n)\mid n=0,1,\cdots,3^2-1\right\}$이 다음과 같음을 알 수 있다.

$$\left\{\frac{0}{9},\frac{3}{9},\frac{6}{9},\frac{1}{9},\frac{4}{9},\frac{7}{9},\frac{2}{9},\frac{5}{9},\frac{8}{9}\right\} \tag{8}$$

식 (5)와 식 (7)을 식 (4.5.9)에 대입하면, $\left\{y^{(2)}(n)=\boldsymbol{\beta}_2^t\boldsymbol{y}_2^{(2)}(n)\mid n=0,1,\cdots,3^2-1\right\}$이 다음과 같음을 알 수 있다.

$$\left\{\frac{0}{9},\frac{3}{9},\frac{6}{9},\frac{4}{9},\frac{7}{9},\frac{1}{9},\frac{8}{9},\frac{2}{9},\frac{5}{9}\right\} \tag{9}$$

식 (6)과 식 (7)을 식 (4.5.9)에 대입하면, $\left\{y^{(3)}(n)=\boldsymbol{\beta}_2^t\boldsymbol{y}_2^{(3)}(n)\mid n=0,1,\cdots,3^2-1\right\}$이 다음과 같음을 알 수 있다.

$$\left\{\frac{0}{9},\frac{3}{9},\frac{6}{9},\frac{7}{9},\frac{1}{9},\frac{4}{9},\frac{5}{9},\frac{8}{9},\frac{2}{9}\right\} \tag{10}$$

식 (8) ∼ 식 (10)에서 알 수 있듯이, 기수를 3으로 하는 3차원 Faure점열은 다음과 같다.

$$\begin{aligned}\left\{[0,0,0],\left[\frac{1}{3},\frac{1}{3},\frac{1}{3}\right],\left[\frac{2}{3},\frac{2}{3},\frac{2}{3}\right],\left[\frac{1}{9},\frac{4}{9},\frac{7}{9}\right],\left[\frac{4}{9},\frac{7}{9},\frac{1}{9}\right],\right.\\ \left.\left[\frac{7}{9},\frac{1}{9},\frac{4}{9}\right],\left[\frac{2}{9},\frac{8}{9},\frac{5}{9}\right],\left[\frac{5}{9},\frac{2}{9},\frac{8}{9}\right],\left[\frac{8}{9},\frac{5}{9},\frac{2}{9}\right]\right\}\end{aligned} \tag{11}$$

지금까지 과정을 수행해서 3차원 Faure점열을 구하는 과정을 수행하기 위해서, 다음 MATLAB프로그램 FaureSequence101.m을 실행하자.

```
% ------------------------------------------------------------------
%  Filename: FaureSequence101.m
%  Generating Faure Sequence
%  Usage: [Amat beta Fseq] = FaureSequence101(9,3,3)
%  Programmed by CBS
%------------------------------------------------------------------
function [Amat beta Fseq] = FaureSequence101(n,d,b);
% INPUTS  : n - maximum sequence index, positive integer
%           d - sequence dimension, positive integer
%           b - sequence base, integer exceeding 1
% Van der Corput Coefficeints a_l (n)
format rat
r = floor(log(n)/log(b))
nn = 0:n-1;
Amat = [ ];
for dum=1:r
    a =  mod(nn,b);
    nn = (nn-a)/b;
    Amat = [ Amat; a ];
end
% beta and C1
beta = [ 1/b ];
for ii = 1:r-1
    beta = [ 1/b ; beta/b ];
end
```

```
C1 = zeros(r,r);
for ii = 1:r
    for jj = ii:r
        C1(ii,jj) = nchoosek(jj-1,ii-1);
    end
end
% Elements of Faure Sequence
Cold = eye(r);
yrdum =  mod(Cold*Amat,b);
Fseq = beta'*yrdum;
for dum=2:d
    Cnew = C1*Cold;
    yrdum = mod(Cnew*Amat,b);
    yr = beta'*yrdum;
    Fseq = [ Fseq ; yr ];
    Cold = Cnew;
end
% End of Program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 사용해서 기수가 3인 차원 Faure점열을 생성하기 위해서 MATLAB커맨드행에서 다음 명령문을 실행하자.

```
>> [Amat beta Fseq] = FaureSequence101(9,3,3)
```

이 MATLAB명령문을 실행한 결과물은 다음과 같다.

```
Amat =
  0    1    2    0    1    2    0    1    2
  0    0    0    1    1    1    2    2    2


beta =
  1/3
  1/9


Fseq =
  0    1/3  2/3  1/9  4/9  7/9  2/9  5/9  8/9
  0    1/3  2/3  4/9  7/9  1/9  8/9  2/9  5/9
  0    1/3  2/3  7/9  1/9  4/9  5/9  8/9  2/9
```

즉, 식 (11)의 Faure점열이 출력된다. ■

Faure점열을 생성하는 점화식을 유도해보자. 식 (4.5.2)에서 알 수 있듯이, $i=1$인 경우에 다음 식들이 성립한다.

$$y_j^{(1)}(n) = a_j(n), \quad (j = 0, 1, \cdots) \tag{4.5.10}$$

식 (4.5.2)와 식 (4.5.3)에서 알 수 있듯이, 각 $i(=2,3,\cdots,d)$와 각 $j(=0,1,\cdots)$에 대해서 다음 식이 성립한다.

$$y_j^{(i)}(n) = \sum_{l=j}^{\infty} \binom{l}{j} \{[i-2]+1\}^{l-j} a_l(n) \pmod b \tag{4.5.11}$$

따라서, 다음 식들이 성립한다.

$$\begin{aligned} y_j^{(i)}(n) &= \sum_{l=j}^{\infty} \sum_{m=0}^{l-j} \binom{l}{j} \binom{l-j}{m} [i-2]^{l-j-m} a_l(n) \pmod b \\ &= \sum_{k=j}^{\infty} \sum_{m=j-k}^{\infty} \binom{m+k}{j} \binom{m+k-j}{m} [i-2]^{k-j} a_{m+k}(n) \pmod b \\ &= \sum_{k=j}^{\infty} \binom{k}{j} \sum_{m=j-k}^{\infty} \binom{m+k}{k} [i-2]^{k-j} a_{m+k}(n) \pmod b \end{aligned} \tag{4.5.12}$$

여기서 첫 번째 등호는 이항정리에 의해서, 두 번째 등호는 변수변환 $k=l-m$에 의해서, 그리고 세 번째 등호는 조합(combination)의 성질에 의해서 성립한다. 식 (4.5.12)의 우변에 변수변환 $p=m+k$를 적용하면, 다음 식이 성립함을 알 수 있다.

$$y_j^{(i)}(n) = \sum_{k=j}^{\infty} \binom{k}{j} \sum_{p=j}^{\infty} \binom{p}{k} [i-2]^{k-j} a_p(n) \pmod b \tag{4.5.13}$$

식 (4.5.2)와 식 (4.5.13)에서 알 수 있듯이, 각 $j(=0,1,\cdots)$에 대해서 다음 점화식이 성립한다.

$$y_j^{(i)}(n) = \sum_{k=j}^{\infty} \binom{k}{j} y_k^{(i-1)}(n) \pmod b, \quad (i=2,3,\cdots,d) \tag{4.5.14}$$

식 (4.5.14)를 식 (4.5.4)에 대입하면, Faure점열을 구할 수 있다.

예제 4.5.2 점화식 (4.5.14)를 사용해서 Faure점열을 생성하기 위해서, 다음 MATLAB프로그램 FaureSequence102.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: FaureSequence102.m
%  Generating Faure Sequence Using Recursive Formula
%  Usage: [Amat beta Fseq] = FaureSequence102(9,3)
%  Programmed by CBS
%------------------------------------------------------------------
function [Amat beta Fseq] = FaureSequence102(Nobs,d)
% Inputs  : Nobs - maximum sequence index, nonnegative integer
%           d    - sequence dimension, positive integer
format rat
% Dertermine Base b
```

```
for dum1 = 0:1000
    if isprime(d+dum1) == 1
       b = d + dum1
       break
    end
end
r = floor(log(Nobs)/log(b)+1)
beta = b.^(-(1:r))
% Radical-Inverse Function
ndum = 0:(Nobs-1);
Amat = [ ]; Fseq = [ ];
for jj=0:r-1
    a = mod(ndum,b);
    ndum = (ndum-a)/b;
    Amat = [ Amat; a ];
end
% Calculating Faure Sequence
for n = 1:Nobs
    y1(:,1) = Amat(:,n);
    yy(1) = beta*y1(:,1);
    % Using Recursive Formula
    for ii=2:d
        for jj=0:r-1
            jj1 = jj+1;
            y1(jj1,ii) = 0;
            for kk = jj:r-1
                y1(jj1,ii) = y1(jj1,ii) + nchoosek(kk,jj) ...
                             *y1(kk+1,ii-1);
            end
            y1(jj1,ii) = mod(y1(jj1,ii),b);
            yy(ii) = beta*y1(:,ii);
        end
    end
    Fseq(n,:) = yy;
end
% End of program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 사용해서 기수가 3인 3차원 Faure점열을 생성하기 위해서 MATLAB커맨드행에서 다음 명령문을 실행하라.

```
>> Fseq = FaureSequence102(9,3)
```

이 MATLAB명령문을 실행한 결과물은 다음과 같다.

```
Amat =

  0    1    2    0    1    2    0    1    2

  0    0    0    1    1    1    2    2    2

  0    0    0    0    0    0    0    0    0


beta =

  1/3   1/9   1/27
```

```
Fseq =
   0     0     0
  1/3   1/3   1/3
  2/3   2/3   2/3
  1/9   4/9   7/9
  4/9   7/9   1/9
  7/9   1/9   4/9
  2/9   8/9   5/9
  5/9   2/9   8/9
  8/9   5/9   2/9
```

즉, 이 MATLAB명령문을 실행하면, 예제 4.5.1과 같은 결과가 출력된다. ■

차수 d가 높은 경우에는 초입방체에서 Faure점열이 Halton점열보다 균등하게 분포한다. 그러나, 일반적으로 d 이상인 소수 중에서 가장 작은 것을 Faure점열의 기수 b로 사용하므로, 차수 d가 아주 높은 경우에는 Faure점열 역시 점들이 뭉쳐지거나 결여되는 현상을 보인다. 좀 더 자세한 내용은 Glasserman [17, p. 301]을 참조하라.

예제 4.5.3 차원이 높은 경우 Faure점열이 균등하게 분포하지 않음을 보이기 위해서, 다음 MATLAB프로그램 FaureSequenceHighDim101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: FaureSequenceHighDim101.m
%  High Dimenional Faure Sequence
%  Programmed by CBS
%------------------------------------------------------------------------
% Construct a d-dimensional point set x of Faure sequence
clear all, close all
n = 1000; d = 19;
[Amat beta fau] = FaureSequence102(n,d);
whos
plot(fau(:,d-1),fau(:,d),'k.','linewidth',1)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf x_{18}')
ylabel('\bf x_{19}','rotation',0)
axis square
saveas(gcf,'FaureSequenceHighDim101','epsc')
save('FaureSequenceHighDim101','fau')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB명령문이 실행되면, 차원 d가 19인 Faure점열을 생성한다. 이 Faure점열의 제18번째 원소 대 제19번째 원소의 산점도가 그림 4.5.1에 그려져 있다. 이 그림에서 알 수

있듯이, 높은 차원의 Faure점열은 초입방체에서 균등하게 분포하기보다는 점들이 뭉쳐지는 현상을 보인다. ■

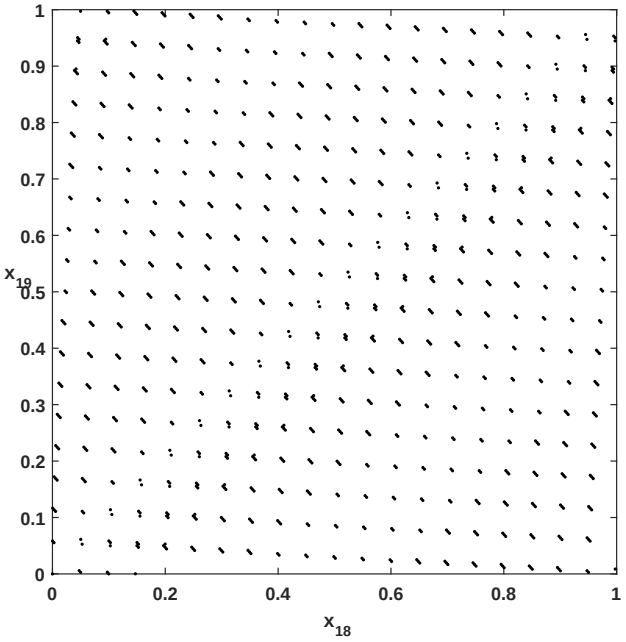


그림 4.5.1. 고차원 Faure점열

예제 4.5.4 잘못 선택된 기수가 좋지 않은 Faure점열을 생성하는 예를 보기 위해서, 다음 MATLAB프로그램 FaureBadBase101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: FaureBadBase101.m
%  Bad Base for Faure Sequence
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all
n = 50
rand('twister',5489)
psuedo = rand(50,2);
[Amat beta Fseq1] = FaureSequence102(50,2);
[Amat beta Fseq2] = FaureSequence102(50,43);
[Amat beta Fseq3] = FaureSequence102(50,109);
subplot(2,2,1)
plot(psuedo(:,1),psuedo(:,2),'k.','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf Psuedo-Random'),ylabel('\bf Psuedo-Random'), grid on
subplot(2,2,2)
plot(Fseq1(:,1),Fseq1(:,2),'k.','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf y_{2}'),ylabel('\bf y_{3}'), grid on
subplot(2,2,3)
plot(Fseq2(:,1),Fseq2(:,2),'k.','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf y_{42}'),ylabel('\bf y_{43}'), grid on
subplot(2,2,4)
```

```
plot(Fseq3(:,1),Fseq3(:,2),'k.','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf y_{108}'),ylabel('\bf Base y_{109}'), grid on
saveas(gcf,'FaureBadBase101','epsc')
save('FaureBadBase101','Fseq1','Fseq1','Fseq3')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 그림 4.5.2이 그려진다. 그림 4.5.2의 좌측상단 그래프는 일양난수들의 산점도이고, 우측상단의 그래프는 기수가 2인 Faure점열의 산점도이다. 일양난수들의 산점도보다는 기수가 2인 Faure점열의 산점도가 좀 더 균등하게 분포하는 것 같다. 좌측하단의 그래프는 기수가 43인 Faure점열의 산점도이고, 우측하단의 그래프는 기수가 109인 Faure점열의 산점도이다. 이 그래프들에서 알 수 있듯이, 기수가 크면 Faure 점열들이 균등하게 발생되지 않는다. ■

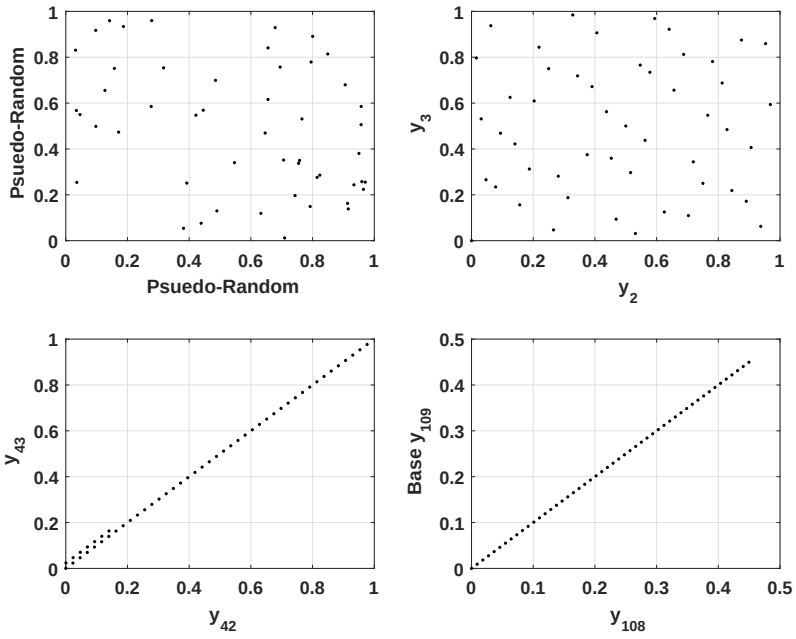


그림 4.5.2. 고차원 Faure점열

예제 4.5.5 예제 4.4.6에서 Halton점열을 사용해서 계산한 2차원 정적분을 Faure점열을 사용해서 구하기 위해서, 다음 MATLAB프로그램 FaureIntegral101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: FaureIntegral101.m
%  Faure 2-D Integration 1
%  Programmed by CBS
%-------------------------------------------------------------------------
clear all, close all, format long
f = @(x,y) exp(-x.*y).*(sin(pi*x)+cos(pi*y));
Tint = dblquad(f,0,1,0,1)
```

```
Mmax = 20;
delta = 100;
rand('twister',5489)
psuedo = rand(Mmax*delta,2);
[Amat beta Fseq] = FaureSequence102(Mmax*delta,2);
for m=1:Mmax
    Pran = f(psuedo(1:m*delta,1),psuedo(1:m*delta,2));
    MC(m) = mean(Pran);
    Qran = f(Fseq(1:m*delta,1),Fseq(1:m*delta,2));
    QMC(m) = mean(Qran);
end
% Plotting
xx = 1:Mmax;
plot([0 Mmax],[Tint Tint],'b-',xx,MC,'k-*', ...
     xx,QMC,'r--o','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xtick',0:5:20)
set(gca,'xticklabel',(0:5:20)*delta)
legend('Value','MC','Faure-QMC','location','SE')
saveas(gcf,'FaureIntegral101','epsc')
save('FaureIntegral101','MC','QMC')
% End of program
% --------------------------------------------------------------------------
```

이 MATLAB프로그램은 다음 정적분을 몬테카를로법과 준몬테카를로법으로 계산하기 위한 것이다.

$$I \doteq \int_0^1 \int_0^1 \exp(-xy)\,[\sin \pi x + \cos \pi y]\,dxdy \tag{1}$$

예제 2.12에서와 마찬가지로 MATLAB함수 dblquad.m을 사용하면, 다음 식이 성립함을 알 수 있다.

$$I = 0.578396241990147 \tag{2}$$

이 MATLAB을 실행하면, 그림 4.5.3 가 그려진다. 그림 4.5.3 에서 흑색 실선은 몬테카를로적분을, 그리고 적색 긴점선은 Faurer점열을 사용한 준몬테카를로적분을 나타낸다. 이 그림에서 알 수 있듯이, 준몬테카를로적분이 몬테카를로적분보다 더 빨리 적분값에 수렴한다. 예제 4.4.6에서 Halton점열을 사용해서 계산한 준몬테카를로적분보다는 Faurer점열을 사용한 준몬테카를로적분이 약간 더 빨리 적분값에 수렴하는 것 같다. ■

제4.6절 Sobol열

Halton점열에서는 큰 소수를 기수로 사용한다. 따라서, 그림 4.4.3에서 볼 수 있듯이, 높은 차원에서 생성된 Halton점열들의 사영(projection)은 군집현상을 보인다. 그림 4.5.1에서 볼 수 있듯이, Faure점열 역시 높은 차원에서는 이러한 군집현상을 보인다. 반면에, Sobol [44]이 제시한 Sobol점열은 단지 2만을 기수로 사용하기 때문에 높은 차원에서 생성된 점열들도

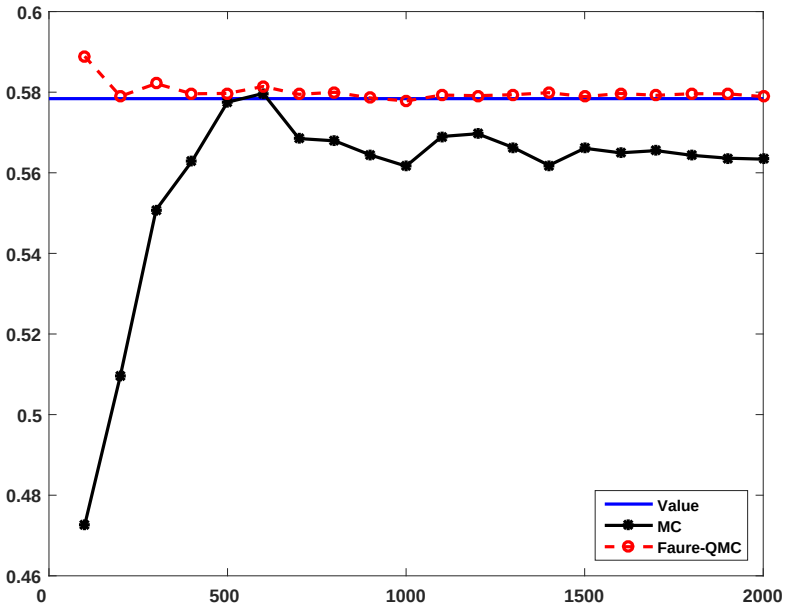


그림 4.5.3. Monte Carlo적분과 Faure적분

균등하게 분포하는 경향이 있다. 또한, 기수로 2를 사용한다는 것은 컴퓨터의 2진구조(binary structure)를 이용할 수 있다는 장점도 있다. Sobol점열에 대한 자세한 내용은 Glasserman [17, 제5.2.3소절]을 참조하고, Sobol점열을 생성하는 프로그램코드에 대해서는 다음 웹사이트를 참조하라.

`http://www.mathfinance.cn/tags/sobol/`

Sobol점열은 기수가 2인 Van der Corput열을 변환해서 생성한다. 즉, Sobol점열의 각 차원의 수열을 다음과 같은 방법으로 구축할 수 있다. 우선, 자연수 n을 다음과 같이 전개하자.

$$n = a_0(n) + a_1(n)2^1 + \cdots + a_{r-1}(n)2^{r-1} \tag{4.6.1}$$

여기서 $a_j(n)$은 0 또는 1이다. Sobol점열 $\{\boldsymbol{x}_n = [x_{n,1}, x_{n,2}, \cdots, x_{n,d}] \mid n = 1, 2, \cdots\}$의 제$n$번째 점 $\boldsymbol{x}_n$의 제j성분 $x_{n,j}$는 다음과 같이 정의된다.

$$x_{n,j} \doteq a_0(n)v_1 \oplus a_1(n)v_2 \oplus a_2(n)v_3 \oplus \cdots \tag{4.6.2}$$

여기서 $v_j \doteq \frac{m_j}{2^j}$이고 m_j는 집합 $\{0, 1, \cdots, 2^j - 1\}$에 속하는 정수이다. 또한, $\oplus$는 다음과 같이 정의되는 2진법의 XOR연산(exclusive or operation)이다.

$$0 \oplus 0 = 0, \quad 0 \oplus 1 = 1, \quad 1 \oplus 0 = 0, \quad 1 \oplus 1 = 0 \tag{4.6.3}$$

식 (4.6.3)의 연산은 컴퓨터에서 실장(**實裝**, implementing)하는데 유용하다. 식 (4.6.2)의 v_j를 방향수(direction number)라 부른다. 따라서, Sobol점열을 이해하기 위해서는 방향수를 이해해야 한다.

방향수를 이해하기 위해서 생성행렬(generator matrix)을 도입하기로 하자. 우선 Sobol 점열의 제j번째 차원에 해당하는 생성행렬을 V_j를 구해보자. 생성행렬 V_j는 각 원소가 0 또는 1인 상삼각행렬이다. 이 생성행렬에 해당하는 방향수들 $v_{1,j}, v_{2,j}, \cdots, v_{r,j}$라고 하자. 여기서 r은 적절한 자연수이다. 그러나, Sobol점열의 제n번째 점을 구축할 때, n의 2진전개(binary expansion)에서 나타나는 항들의 수를 r로 간주한다. 즉, 다음과 같은 r을 사용한다.

$$r = \left\lfloor \frac{\log_2 n}{\log_2 p} \right\rfloor + 1 \tag{4.6.4}$$

제k번째 방향수 $v_{k,j}$를 다음과 같이 2진전개하자.

$$v_{k,j} = \frac{\delta_{k,1}}{2^1} + \frac{\delta_{k,2}}{2^2} + \cdots + \frac{\delta_{k,r}}{2^r} \tag{4.6.5}$$

벡터 $[\delta_{1,k}, \delta_{2,k}, \cdots, \delta_{r,k}]^t$를 생성행렬 V_j의 제k번째 열벡터라 하자. 즉, 다음 식이 성립한다.

$$V_j \doteq \begin{bmatrix} \delta_{1,1} & \delta_{1,2} & \cdots & \delta_{1,r} \\ \delta_{2,1} & \delta_{2,2} & \cdots & \delta_{2,r} \\ \vdots & \vdots & & \vdots \\ \delta_{r,1} & \delta_{r,2} & \cdots & \delta_{r,r} \end{bmatrix} \tag{4.6.6}$$

다음 벡터를 정의하자.

$$\boldsymbol{v}_k \doteq [\delta_{1,k}, \delta_{2,k}, \cdots, \delta_{r,k}]^t \tag{4.6.7}$$

따라서 다음 식이 성립한다.

$$V_j = [\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_k] \tag{4.6.8}$$

따라서, 상삼각행렬 V_j는 $r \times r$ 행렬이며, 방향수벡터 $[v_{1,j}, v_{2,j}, \cdots, v_{r,j}]$와 생성행렬 V_j는 1대1 대응관계에 있다. 이 방향수들의 선택에 대해서는 뒤에서 다루게 될 것이다.

벡터 $\boldsymbol{a}(n) \doteq [a_0(n), a_1(n), \cdots, a_{r-1}(n)]^t$과 생성행렬 V_j에 대해서 다음과 같은 벡터 $\boldsymbol{y}(n) \doteq [y_1(n), y_2(n), \cdots, y_r(n)]^t$을 정의하자.

$$\boldsymbol{y}(n) \doteq V_j \boldsymbol{a}(n) \pmod 2 \tag{4.6.9}$$

식 (4.6.2)와 식 (4.6.9)에서 알 수 있듯이, Sobol점열 $\{\boldsymbol{x}_n \mid n = 1, 2, \cdots\}$의 제$n$번째 점 $\boldsymbol{x}_n$의 제j번째 성분 $x_{n,j}$는 다음과 같다.

$$x_{n,j} \doteq \frac{y_1(n)}{2} + \frac{y_2(n)}{2^2} + \cdots + \frac{y_r(n)}{2^r} \tag{4.6.10}$$

즉, 벡터 $\boldsymbol{y}(n)$은 2진수로 표시된 것이고, $x_{n,j}$는 이를 십진수로 나타낸 것이다. 만약 V_j가 단위행렬이면, $[y_1(n)\ y_2(n)\ \cdots\ y_r(n)]$은 기수가 2인 Van der Corput열이다. 식 (4.6.9)을 다음과 같이 쓸 수 있다.

$$\boldsymbol{y}(n) = a_0(n)\boldsymbol{v}_1 \oplus a_1(n)\boldsymbol{v}_2 \oplus \cdots a_{r-1}(n)\boldsymbol{v}_r \tag{4.6.11}$$

식 (4.6.5)에서 알 수 있듯이 $v_{k,j}$가 2진수이므로, XOR연산 $\oplus$를 사용해서 식 (4.6.11)의 2진수와 이에 해당하는 Sobol점열의 제n번째 점 $\boldsymbol{x}_n$의 제j번째 성분 $x_{n,j}$를 쉽게 계산할 수 있다.

우리의 문제는 제j번째 생성행렬 V_j를 어떻게 선택하는냐는 것이다. 유한체(finite field)란 차수(field order)가 유한인, 즉 원소들의 개수가 유한인 체(field)로서, Galois체라고도 부른다. 유한체의 차수는 소수(prime number)이거나 소수의 멱함수이다. Sobol법에서는 차수가 2인 Galois체 GF(2)에서 정의되는 원시멱함수(primitive polynomial)를 이용해서 생성행렬들을 선택한다. 이 함수는 각 계수가 0 또는 1인 q차 멱함수로서 다음 두 조건들을 만족한다. 첫째 이 멱함수는 기약적(irreducible)이고, 둘째 이 멱함수가 $x^p + 1$으로 나누어지는 가장 작은 차수는 $p = 2^q - 1$인 경우이다. 다음과 같은 함수들이 원시멱함수이다.

$$P_1(x) = x + 1,\ \ P_2(x) = x^2 + x + 1,\ \ P_3(x) = x^3 + x + 1,\ \ P_4(x) = x^3 + x^2 + 1 \tag{4.6.12}$$

Sobol점열의 제j번째 생성행렬 V_j를 선택하기 위해서, 먼저 다음과 같이 Galois체 GF(2)에서 차수가 s_j인 원시멱함수를 선택한다.

$$P_j(x) = x^{s_j} + p_{1,j}x^{s_j-1} + p_{2,j}x^{s_j-2} + \cdots + p_{s_j-1,j}x^1 + 1 \tag{4.6.13}$$

여기서 $p_{1,j}, p_{2,j}, \cdots, p_{s_j-1,j}$는 0 또는 1이다. Sobol점열의 오차한계를 작게 하기 위해서는 되도록 찻수가 낮은 원시멱함수를 선택해야 한다. 다음 점화식을 사용해서 자연수열

$\{m_{1,j}, m_{2,j}, \cdots\}$를 정의하자.

$$m_{k,j} = 2p_{1,j}m_{k-1,j} \oplus \cdots \oplus 2^{s_j-1}p_{s_j-1,j}m_{k-s_j+1,j} \oplus 2^{s_j}m_{k-s_j,j} \oplus m_{k-s_j,j} \tag{4.6.14}$$

여기서 각 $k(=1,2,\cdots,s_j)$에 대한 초기값 $m_{k,j}$는 2^k보다 작은 임의의 홀수를 선택한다. 어떤 초기값들을 선택하느냐에 따라, 생성된 Sobol점열의 질이 달라진다. 이에 대한 자세한 내용은 Jäckel [18, p. 87]과 Joe & Kuo [19]을 참조하라. 점화식 (4.6.14)에 의해서 생성된 자연수열 $\{m_{1,j}, m_{2,j}, \cdots, m_{r,j}\}$를 사용해서, 다음과 같이 방향수들 $v_{1,j}, v_{2,j}, \cdots, v_{r,j}$를 정의한다.

$$v_{k,j} \doteq \frac{m_{k,j}}{2^k}, \quad (k = 1,2,\cdots,r) \tag{4.6.15}$$

점화식 (4.6.14)을 사용해서 방향수들과 생성행렬을 구해보자.

예제 4.6.1 이 예제는 Glasserman [17, pp. 306-307]에서 인용한 것이다. 멱함수 $P(x) = x^3 + x^2 + 1$는 원시멱함수이다. 이 경우에, 식 (4.6.14)를 다음과 같이 쓸 수 있다.

$$m_k = 2m_{k-1} \oplus 8m_{k-3} \oplus m_{k-3}, \quad (k = 4,5,\cdots) \tag{1}$$

수열 $\{m_k\}$의 초기값들로 다음 값들을 사용하자.

$$m_1 = 1 = 1_2, \quad m_2 = 3 = 11_2, \quad m_3 = 3 = 11_2 \tag{2}$$

식 (2)를 식 (1)에 대입하면, 다음 식들을 얻는다.

$$\begin{aligned} m_4 &= [2 \cdot 3] \oplus [8 \cdot 1] \oplus 1 = 0110_2 \oplus [1000_2 \oplus 0001_2] \\ &= 0110_2 \oplus 1001_2 = 1111_2 = 15 \end{aligned} \tag{3}$$

$$\begin{aligned} m_5 &= [2 \cdot 15] \oplus [8 \cdot 3] \oplus 3 = 11110_2 \oplus [11000_2 \oplus 00011_2] \\ &= 11110_2 \oplus 11011_2 = 00101_2 = 5 \end{aligned} \tag{4}$$

따라서, 방향수들은 다음과 같다.

$$v_1 = 0.10000_2, \quad v_2 = 0.11000_2, \quad v_3 = 0.01100_2, \quad v_4 = 0.11110_2, \quad v_5 = 0.00101_2 \tag{5}$$

이에 대응하는 발생행렬 V 의 제j 번째 열은 방향수 v_j 의 2진표현을 열벡터로 나타낸 것이므로, 다음 식이 성립한다.

$$V = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

만약 초기값들을 $m_1 = m_2 = m_3 = 1$ 으로 선택하면, 점화식 (1)은 다음과 같다.

$$m_k = m_{k-3} = 1, \quad (k = 4, 5, \cdots) \tag{7}$$

따라서, 방향수들은 다음과 같다.

$$v_k = \frac{m_k}{2^k} = \frac{1}{2^k}, \quad (k = 1, 2, \cdots) \tag{8}$$

이에 해당하는 발생행렬 V 는 단위행렬이다. ■

예제 4.6.2 이 예제는 Bratley & Fox [11]에서 인용한 것이다. 원시멱함수 $P(x) = x^3 + x + 1$ 에 대해서 식 (4.6.14)을 다음과 같이 쓸 수 있다.

$$m_k = 4m_{k-2} \oplus 8m_{k-3} \oplus m_{k-3} \tag{1}$$

수열 $\{m_k\}$ 의 초기값들로 다음 값들을 사용하자.

$$m_1 = 1 = 1_2, \quad m_2 = 3 = 11_2, \quad m_3 = 7 = 111_2 \tag{2}$$

예제 4.6.1에서와 같이, 식 (2)를 식 (1)에 대입하면, 다음 식들을 얻는다.

$$m_4 = 4m_2 \oplus 8m_1 \oplus m_1 = 4m_2 \oplus [8m_1 \oplus m_1] = 5 = 101_2 \tag{3}$$

$$m_5 = 4m_3 \oplus 8m_2 \oplus m_2 = 4m_3 \oplus [8m_2 \oplus m_2] = 7 = 111_2 \tag{4}$$

$$m_6 = 4m_4 \oplus 8m_3 \oplus m_3 = 4m_4 \oplus [8m_3 \oplus m_3] = 43 = 110101_2 \tag{5}$$

따라서, 방향수들은 다음과 같다.

$$v_1 = 0.100000_2, \quad v_2 = 0.110000_2, \quad v_3 = 0.111000_2 \tag{6}$$

$$v_4 = 0.101000_2, \quad v_5 = 0.111000_2, \quad v_6 = 0.110101_2 \tag{7}$$

이에 대응하는 발생행렬 V는 다음과 같다.

$$V = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

MATLAB함수 bitxor.m를 사용해서 식 (3) ∼ 식 (7)을 계산할 수 있다. 다음 MATLAB 프로그램 DirectionNumbers101.m을 실행해보자.

```
% ------------------------------------------------------------------
%  Filename: DirectionNumbers101.m
%  Generating Direction Numbers from P(x) = x^3 + x + 1
%  Programmed by CBS
%------------------------------------------------------------------
diary DirectionNumbers101.txt
clear all, close all, format rat
m = [1 3 7 ];
k = 6
for kk = 4:k
    m(kk) = bitxor(4*m(kk-2),bitxor(8*m(kk-3),m(kk-3)));
end
disp('m = '), disp(m)
v = m./(2.^(1:length(m)))
Vmat = zeros(k,k);
for kk=1:k
    dumR = v(kk);
    for ii=1:k
        Vmat(ii,kk) = floor(dumR*2);
        dumR = dumR*2 - Vmat(ii,kk);
    end
end
Vmat
diary off
% End of Program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 다음 결과를 얻는다.

```
k =
 6

m =
 1     3     7     5     7    43

v =
 1/2   3/4   7/8   5/16  7/32    43/64

Vmat =
 1     1     1     0     0     1
 0     1     1     1     0     0
 0     0     1     0     1     1
 0     0     0     1     1     0
 0     0     0     0     1     1
 0     0     0     0     0     1
```

■

예제 4.6.3 일반적인 원시멱함수에 대한 방향수들을 찾기 위해서, 다음 MATLAB프로그램 DirectionNumbers102.m를 사용할 수 있다.

```
% ------------------------------------------------------------------------
%  Filename: DirectionNumbers102.m
%  Generating Direction Numbers
%     from P(x) = p_0*x^s + p_1*x^s-1 + ... + p_s-1*x^1 + p_s*x^0
%  Usage: >> P = [ 1 0 1 1 ], m0 = [ 1 3 7], r = 6
%         >> [m Vmat v] =  DirectionNumbers102(P,m0,r)
%  Programmed by CBS
%------------------------------------------------------------------------
function [m,mBIN,v] = DirectionNumbers102(P,mInit,r)
format rat
% INPUTS  : P = [p_0 p_1 ... p_s ] - Primitive Polynomial
%           mInit - Initial Values of m
%           r - Number of directional numbers
deg = length(P) -1;    % Degree of the primitive polynomial
p = P(2:deg);
m = [ mInit, zeros(1,r-deg) ];
for kk = (deg+1):r
    m(kk) = bitxor(m(kk-deg),2^deg*m(kk-deg));
    for ii = 1:(deg-1)
        m(kk) = bitxor(m(kk),2^ii*p(ii)*m(kk-ii));
    end
end
mBIN = de2bi(m)';
v = m./(2.^(1:length(m)));
```

```
end
% End of Program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 사용해서 원시멱함수 $P(x) = x^3 + x + 1$에 대한 방향수들을 생성하기 위해서, MATLAB커맨드행에서 다음 명령문들을 실행하라.

```
>> P = [ 1 0 1 1 ], m0 = [ 1 3 7], r = 6
>> [m Vmat v ] =  DirectionNumbers102(P,m0,r)
```

이 MATLAB프로그램을 실행하면, 다음 결과를 얻는다.

```
r =    6


m =     1             3             7             5             7            43


Vmat =
       1             1             1             1             1             1
       0             1             1             0             1             1
       0             0             1             1             1             0
       0             0             0             0             0             1
       0             0             0             0             0             0
       0             0             0             0             0             1


v =   1/2           3/4           7/8          5/16          7/32          43/64
```

이 결과는 예제 4.6.2의 결과와 동일하다. ■

예제 4.6.4 식 (4.6.11)을 사용해서 Sobol점열을 구해보자. 이 예제는 Glasserman [17, pp. 306-307]에서 인용한 것이다. 예제 4.6.1에서와 같은 방법을 적용하면, 원시멱함수

$P(x) = x^3 + x + 1$에 대해서 방향수들과 발생행렬 V는 각각 다음과 같음을 알 수 있다.

$$v_1 = 0.100000_2, \quad v_2 = 0.110000_2, \quad v_3 = 0.011000_2 \tag{1}$$

$$v_4 = 0.111100_2, \quad v_5 = 0.001010_2, \quad v_6 = 0.010001_2 \tag{2}$$

$$V = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

또한, 기수가 2인 근기역함수는 다음과 같다.

$$\psi_2(1) = 0.100000_2, \quad \psi_2(2) = 0.010000_2, \quad \psi_2(3) = 0.110000_2 \tag{4}$$

$$\psi_2(4) = 0.001000_2, \quad \psi_2(5) = 0.101000_2, \quad \psi_2(6) = 0.011000_2 \tag{5}$$

즉, 다음 식이 성립한다.

$$A \doteq \begin{bmatrix} a_0(1) & a_0(2) & a_0(3) & a_0(4) & a_0(5) & a_0(6) \\ a_1(1) & a_1(2) & a_1(3) & a_1(4) & a_1(5) & a_1(6) \\ a_2(1) & a_2(2) & a_2(3) & a_2(4) & a_2(5) & a_2(6) \\ a_3(1) & a_3(2) & a_3(3) & a_3(4) & a_3(5) & a_3(6) \\ a_4(1) & a_4(2) & a_4(3) & a_4(4) & a_4(5) & a_4(6) \\ a_5(1) & a_5(2) & a_5(3) & a_5(4) & a_5(5) & a_5(6) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6}$$

식 (4.6.9)에서 알 수 있듯이, 다음 식들이 성립한다.

$$[\boldsymbol{y}(1), \boldsymbol{y}(2), \cdots, \boldsymbol{y}(r)] \doteq VA \pmod 2 = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{7}$$

따라서, 원시멱함수 $P(x) = x^3 + x + 1$에 대한 Sobol점열은 다음과 같다.

$$\left\{\frac{1}{2}, \frac{3}{4}, \frac{1}{4}, \frac{3}{8}, \frac{7}{8}, \frac{5}{8}\right\} \tag{8}$$

식 (3) ∼ 식 (8)을 확인하기 위해서, 다음 MATLAB프로그램 SobolSequenceGen101.m을 실행해보자.

```
% ------------------------------------------------------------------
%  Filename: SobolSequenceGen101.m
%  Generating Sobol Sequece using P(x) = x^3 + x^2 + 1
%  Programmed by CBS
%------------------------------------------------------------------
diary SobolSequenceGen101.txt
clear all, close all, format rat
k = 6    % Length of Sobol sequence
bb = (2.^-(1:k));
r = floor(log(k)/log(2)+1)
% Genetaing Matrix for P(x) = x^3 + x^2 + 1
m = [1 3 3 ];
for kk = 4:k
    m(kk) = bitxor(2*m(kk-1),bitxor(8*m(kk-3),m(kk-3)));
end
disp('m = '), disp(m)
v = m.*bb
Vmat = zeros(k,k);
for kk=1:k
    dumR = v(kk);
    for ii=1:k
        Vmat(ii,kk) = floor(dumR*2);
        dumR = dumR*2 - Vmat(ii,kk);
    end
end
Vmat
% Generating { a_l (n) }
Amat = zeros(k,k);  % The n-th raw = [ a_0 (n) ... a_r-1 (n) ]'
for n = 1:k
    dum = de2bi(n);
    Amat(1:length(dum),n) = dum';
end
disp('Amat = '), disp(Amat)
% Sobol Sequence
SobolBin = mod(Vmat*Amat,2)
Sseq = bb*SobolBin              % Sobol sequence in Digit
diary off
% End of Program
% ------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 다음과 같은 결과를 얻는다.

```
k =

     6


r =
```

```
 3


m =
 1     3     3    15     5    17


v =
 1/2   3/4   3/8  15/16  5/32    17/64


Vmat =
 1     1     0     1     0     0
 0     1     1     1     0     1
 0     0     1     1     1     0
 0     0     0     1     0     0
 0     0     0     0     1     0
 0     0     0     0     0     1


Amat =
 1     0     1     0     1     0
 0     1     1     0     0     1
 0     0     0     1     1     1
 0     0     0     0     0     0
 0     0     0     0     0     0
 0     0     0     0     0     0


SobolBin =
 1     1     0     0     1     1
 0     1     1     1     1     0
 0     0     0     1     1     1
 0     0     0     0     0     0
 0     0     0     0     0     0
 0     0     0     0     0     0


Sseq =
 1/2   3/4   1/4   3/8   7/8   5/8
```

■

Antonov & Saleev [8]는 2진법표현인 $\boldsymbol{a}(n)$ 대신에 Gray코드표현을 사용해서, Sobol방법을 단순화하는 방법을 제시했다. 즉, 식 (4.6.11)을 사용하는 대신에 Gray코드를 바탕으로 하는 방법을 제시하고, 이렇게 생성된 점열도 디스크레펀시를 증가시키지 않음을 보였다. 자연수 $k = [a_{r-1}(k) \cdots a_1(k)a_0(k)]_2$의 Gray코드 $g(k)$는 다음과 같다.

$$g(k) \doteq [a_{r-1}(k) \cdots a_1(k)a_0(k)]_2 \oplus [0a_{r-1}(k) \cdots a_1(k))]_2 \tag{4.6.16}$$

즉, Gray코드 $g(k)$는 k와 $\lfloor k/2 \rfloor$를 XOR연산 $\oplus$를 한 것이다.

예제 4.6.5 식 (4.6.16)을 사용해서, 다음 표의 Gray코드를 계산해보자.

10진법	1	2	3	4	5	6	7	8
2진법	0001	0010	0011	0100	0101	0110	0111	1000

다음 식들이 성립한다.

$$g(1) = 001_2 \oplus 000_2 = 001_2 \tag{1}$$

$$g(2) = 010_2 \oplus 001_2 = 011_2 \tag{2}$$

$$g(3) = 011_2 \oplus 001_2 = 010_2 \tag{3}$$

$$g(4) = 100_2 \oplus 010_2 = 110_2 \tag{4}$$

$$g(5) = 101_2 \oplus 010_2 = 111_2 \tag{5}$$

$$g(6) = 110_2 \oplus 011_2 = 101_2 \tag{6}$$

$$g(7) = 111_2 \oplus 011_2 = 100_2 \tag{7}$$

따라서, 다음 표가 만들어진다.

10진법	1	2	3	4	5	6	7
2진법	001	010	011	100	101	110	111
Gray코드	001	011	010	110	111	101	100

이러한 과정을 수행하기 위해서, 다음 MATLAB프로그램 GrayCode101.m을 사용할 수 있다.

```
% ------------------------------------------------------------------
%  Filename: GrayCode101.m
%  Gray Code for j = 1,2, ..., (2^m -1)
%  Usage: >> [ Gcode ] = GrayCode101(3)
%  Programmed by CBS
%------------------------------------------------------------------
function [ Gcodes] = GrayCode101(m)
format rat
diary GrayCode101.txt
GrayFtn = inline('bitxor(a,bitshift(a,-1))');
Gcodes = zeros(2^m-1,m);
for jj = 1:2^m-1
    Gcodes(jj,:) = bitget(GrayFtn(jj),m:-1:1);
end
diary off
% End of program
% ------------------------------------------------------------------
```

MATLAB커맨드행에서 다음 명령문을 실행하면, 위 표와 같은 Gray코드를 얻는다.

```
>> [ Gcode ] = GrayCode101(3)
```

이 MATLAB명령문을 실행한 결과는 다음과 같다.

```
Gcode =

     0     0     1
     0     1     1
     0     1     0
     1     1     0
     1     1     1
     1     0     1
     1     0     0
```

■

예제 4.6.6 Gray코드를 계산하는 다른 방법은 k의 2진수에서 가장 오른쪽에 위치한 0의 자리에서 $g(k)$의 2진수를 바꾸는 것이다.

예제 4.6.5의 두 번째 표는 다음과 같다.

10진법	1	2	3	4	5	6	7
2진법	001	010	011	100	101	110	111
Gray코드	001	011	010	110	111	101	100

첫째, 1의 2진수 001_2에서 가장 오른쪽의 0은 2^1 자리이다. 따라서, $g(1) = 001_2$에서 2^1 자리의 수가 1로 바뀐다. 즉, $g(2) = 011_2$이다. 둘째, 2의 2진수 010_2에서 가장 오른쪽의 0은 2^0 자리이다. 따라서, $g(2) = 011_2$에서 2^0 자리의 수가 0으로 바뀐다. 즉, $g(3) = 010_2$이다. 셋째, 3의 2진수 011_2에서 가장 오른쪽의 0은 2^2 자리이다. 따라서, $g(3) = 010_2$에서 2^2 자리의 수가 1로 바뀐다. 즉, $g(4) = 110_2$이다. 넷째, 4의 2진수 100_2에서 가장 오른쪽의 0은 2^0 자리이다. 따라서, $g(4) = 110_2$에서 2^0 자리의 수가 1로 바뀐다. 즉, $g(5) = 111_2$이다. 다섯째, 5의 2진수 101_2에서 가장 오른쪽의 0은 2^1 자리이다. 따라서, $g(5) = 111_2$에서 2^1 자리의 수가 0으로 바뀐다. 즉, $g(6) = 101_2$이다. 여섯째, 6의 2진수 110_2에서 가장 오른쪽의 0은 2^0 자리이다. 따라서, $g(6) = 101_2$에서 2^0 자리의 수가 0으로 바뀐다. 즉, $g(7) = 100_2$이다. 만약 $g(8)$을 구하고 싶다면, 다음 표를 가지고 계산한다.

10진법	1	2	3	4	5	6	7	8	9
2진법	00001	00010	00011	00100	00101	00110	00111	01000	01001

이러한 과정을 수행하기 위해서, 다음 MATLAB프로그램 GrayCode102.m을 사용할 수 있다.

```
% ------------------------------------------------------------------
%  Filename: GrayCode102.m
%  Gray Code for j = 1,2, ..., (2^m -1)
%  Programmed by CBS
%------------------------------------------------------------------
function [ cTrans Gcodes2 ] = GrayCode102(m)
format rat
diary GrayCode102.txt
% Find c for j = 1,2, ..., (2^m -1)
c = zeros(2^m -1,1);
for jj = 1:2^m -1
    c(jj,:) = find( bitget(jj,1:2^m) == 0,1,'first');
end
cTrans = c';                % Index of the rightmost zero bit
% Calculate Gray Code
Gcodes10 = zeros(2^m-1,1);
Gcodes10(1,:) = 1;
Gcodes2(1,:) = bitget(Gcodes10(1),m:-1:1);
for jj = 2:2^m-1
    Gcodes10(jj,:) = bitxor(Gcodes10(jj-1,:),2^(c(jj-1)-1));
    Gcodes2(jj,:) = bitget(Gcodes10(jj),m:-1:1);
end
diary off
% End of program
% ------------------------------------------------------------------
```

MATLAB커맨드행에서 다음 명령문을 실행하자.

```
>> [ cTrans Gcode2 ] = GrayCode102(3)
```

이 MATLAB명령문을 실행한 결과는 다음과 같다.

```
cTrans =
    2     1     3     1     2     1     4
Gcodes2 =
    0     0     1
    0     1     1
    0     1     0
    1     1     0
    1     1     1
    1     0     1
    1     0     0
```

■

Gray코드 $g(k)$를 다음과 같이 표현하자.

$$g(k) = \sum_{l=0}^{r-1} g_l(k) 2^l \tag{4.6.17}$$

예제 4.6.5에서 알 수 있듯이, 임의의 자연수 r에 대해서 집합 $\{0, 1, 2, \cdots, 2^r - 1\}$의 Gray코드에 의한 2진스트링들(binary strings)은 이 집합의 원소들을 보통 2진법에 의해 나타낸 2진스트링들의 순열이다. 따라서, 근기역함수 $\psi_2(k) = \sum_{l=0}^{r-1} a_l(k) 2^{-l-1}$에서 $a_l(k)$를 $g_l(k)$로 바꾼 $\hat{\psi}_2(k) \doteq \sum_{l=0}^{r-1} g_l(k) 2^{-l-1}$들의 집합 $\{\hat{\psi}_2(k) \mid k = 0, 1, \cdots, 2^r - 1\}$은 Van der Corput열 $\{\psi_2(k) \mid k = 0, 1, \cdots, 2^r - 1\}$의 순열이다. Antanov & Saleev [8]는 이렇게 생성된 $\{\hat{\psi}_2(k)\}$와 Van der Corput열 $\{\psi_2(k)\}$의 디스크레펀시는 점근적으로 같음을 증명하였다.

Gray코드를 사용해서 변형된 Sobol점열을 생성하는 가장 큰 이유는 다음과 같이 간단하게 점열들을 계산할 수 있는 점화식이 성립하기 때문이다. 식 (4.6.11)의 Sobol점열에서 $a_l(k)$를 $g_l(k)$로 바꾼 $\hat{\boldsymbol{y}}(k)$가 Gray코드에 의한 Sobol점의 2진표현이다. 즉, 다음 식이 성립한다.

$$\hat{\boldsymbol{y}}(k) = g_0(k)\boldsymbol{v}_1 \oplus g_1(k)\boldsymbol{v}_2 \oplus \cdots \oplus g_{r-1}(k)\boldsymbol{v}_r \tag{4.6.18}$$

만약 k와 $k+1$의 Gray코드들이 제c번째 bit에서만 다르다면, 다음 식들이 성립한다.

$$\begin{aligned}\hat{\boldsymbol{y}}(k+1) &= g_0(k+1)\boldsymbol{v}_1 \oplus g_1(k+1)\boldsymbol{v}_2 \oplus \cdots \oplus g_{r-1}(k+1)\boldsymbol{v}_r \\ &= g_0(k)\boldsymbol{v}_1 \oplus \cdots \oplus g_{c-1}(k)\boldsymbol{v}_{c-1} \oplus [g_c(k) \oplus 1]\,\boldsymbol{v}_c \oplus g_{c+1}(k)\boldsymbol{v}_{c+1} \oplus \cdots \oplus g_{r-1}(k)\boldsymbol{v}_r \\ &= \hat{\boldsymbol{y}}(k) \oplus \boldsymbol{v}_c \end{aligned} \tag{4.6.19}$$

식 (4.6.3)과 예제 4.6.6에서 알 수 있듯이, $\boldsymbol{v}_c$는 k를 2진수로 표현했을 때 가장 우측에 위치하는 비트 $a_{c-1}(k)$의 c에 해당하는 열벡터이다.

예제 4.6.7 식 (4.6.19)을 사용해서 Sobol점열을 생성하기 위해서, 다음 MATLAB프로그램 SobolSequence101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: SobolSequence101.m
%  Generating Sobol Sequence with Generating Polynomial
%       P(x) = p_0*x^s + p_1*x^s-1 + ... + p_s-1*x^1 + p_s*x^0
% Usages: P = [ 1 0 1 1 ], m0 = [ 1 3 7 ], r = 6
%         [ c Sseq ] = SobolSequence101(P,m0,12)
%  Programmed by CBS
%------------------------------------------------------------------------
function [ c Sseq ] = SobolSequence1(P,mInit,Nseq)
% INPUTS  : P = [p_0 p_1 ... p_s ] - Primitive Polynomial
%           mInit - Initial Values of m
%           Nseq - Number of Points
% Directional Numbers
format rat
r = floor(log(Nseq)/log(2)+1)
deg = length(P) -1;    % Degree of the primitive polynomial
p = P(2:deg);
m = [ mInit, zeros(1,r-deg) ];
for kk = (deg+1):r
    m(kk) = bitxor(m(kk-deg),2^deg*m(kk-deg));
    for ii = 1:(deg-1)
        m(kk) = bitxor(m(kk),2^ii*p(ii)*m(kk-ii));
    end
end
mBin = de2bi(m)';
v = m./(2.^(1:length(m)));
% Find c
for kk=0:Nseq-1
    c(kk+1) = find( bitget(kk,1:2^length(m)) == 0,1,'first');
end
% Generating Sobol Sequence
x0 = 0;                          % Initial Number
Deno = 2^length(m);
Sseq = zeros(Nseq,1);
BitNum = v*Deno;
Sseq(1) = fix(x0*Deno);
for kk=1:Nseq-1
    Sseq(kk+1) = bitxor(Sseq(kk),BitNum(c(kk)));
end
Sseq = Sseq'/Deno;
```

```
% End of Program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 사용해서 원시멱함수 $P(x) = x^3 + x + 1$에 대한 방향수들을 생성하기 위해서, MATLAB커맨드행에서 다음 명령문들을 실행하라.

```
>> P = [ 1 0 1 1 ], m0 = [ 1 3 7 ], r = 6
>> [ c Sseq ] = SobolSequence101(P,m0,12)
```

이 명령문은 실행한 결과는 다음과 같다.

```
c =
 Columns 1 through 6
    1       2       1       3       1       2
 Columns 7 through 12
    1       4       1       2       1       3


Sseq =
 Columns 1 through 6
    0       1/2      1/4      3/4      1/8      5/8
 Columns 7 through 12
    3/8      7/8      11/16      3/16      15/16      7/16
```

■

MATLAB함수 sobolset.m을 사용해서 Sobol점열을 생성할 수 있다. 다음 예제를 살펴보자.

예제 4.6.8 Sobol점열을 생성하기 위해서, 다음 MATLAB프로그램 SobolSequence102.m을 실행해 보자.

```
% ------------------------------------------------------------------
%  Filename: SobolSequence102.m
%  Generating Sobol Sequence
%------------------------------------------------------------------
clear all, close all, format short
Sseq1 = sobolset(3,'Skip',2000,'Leap',10)
Sseq2 = scramble(Sseq1,'MatousekAffineOwen')
Sseq3 = net(Sseq2,4)    % net: Intrinsic function
% End of program
% ------------------------------------------------------------------
```

첫째, sobolset명령문은 차원이 $d = 3$인 Sobol점열을 생성하면서, 첫 2000개는 버리고(skip), 다음으로 10개 점들을 버리고(leap) 11번째 점을 남기는 과정을 반복하도록 한다. 이 sobolset명령문을 실행한 결과는 다음과 같다.

```
Sseq1 =
Sobol point set in 3 dimensions (818836295885363 points)
Properties:
              Skip : 2000
              Leap : 10
    ScrambleMethod : none
        PointOrder : standard
```

둘째, scramble명령문은 생성된 Sobol점열 Sseq1에 무작위디지탈이동(random digital shift)을 이용한 무작위선형섞음(random linear scramble)을 실행한다. 이 scramble명령문을 실행한 결과는 다음과 같다.

```
Sseq2 =
Sobol point set in 3 dimensions (818836295885363 points)
Properties:
              Skip : 2000
              Leap : 10
    ScrambleMethod : MatousekAffineOwen
        PointOrder : standard
```

셋째, net명령문은 Sobol점열 Sseq2를 출력하기 위한 것이다. Halton점열의 경우와 마찬가지로, MATLAB명령문들 sobelset.m이나 scramble.m을 실행했다고 해서 Sobol점열이 만들어 지는 것은 아니다. Sobol점열을 생성하기 위해서는 MATLAB함수 net.m을 사용하거나, 행렬 Sseq2에서 출력하고자 하는 원소들의 첨자들을 지정한다. 이 net명령문을 실행한 결과는 다음과 같다.

```
Sseq3 =
    0.5072    0.2542    0.5177
    0.1417    0.5408    0.7887
    0.8922    0.4131    0.6685
    0.4965    0.2055    0.8025
```

■

예제 4.6.9 차원이 높은 경우 Sobol점열이 Halton점열보다는 더 균등하게 분포함을 보이기 위해서, 다음 MATLAB프로그램 SobolSequenceHighDim101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: SobolSequenceHighDim101.m
%  High Dimenional Sobol Sequence
%  Programmed by CBS
%-------------------------------------------------------------------------
% Construct a d-dimensional point set x of Sobol sequence
n = 1000; d = 19;
x = sobolset(d)
sob = x(1:1:n,:);
plot(sob(:,d-1),sob(:,d),'k.','linewidth',1)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf x_{18}')
ylabel('\bf x_{19}','rotation',0)
axis square
saveas(gcf,'SobolSequenceHighDim101','epsc')
save('SobolSequenceHighDim101','sob')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB명령문이 실행되면, 차원 d가 19인 Sobol점열을 생성한다. 이 Sobol점열의 18번째 원소 대 19번째 원소의 산점도가 그림 4.6.1에 그려져 있다. 이 그림을 그림 4.4.3이나 그림 4.5.1과 비교하면, 높은 차원의 Sobol점열은 Halton점열이나 Faure점열보다 초입방체에서 균등하게 분포하는 경향이 있음을 알 수 있다. ■

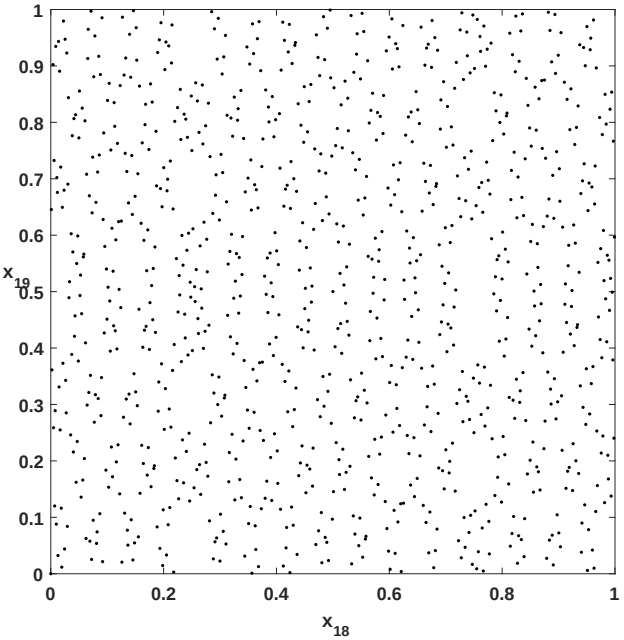


그림 4.6.1. 고차원 Sobol점열

예제 4.6.10 예제 4.4.6와 예제 4.5.5에서 계산한 2차원 정적분을 Sobol점열을 사용해서 구하기 위해서, 다음 MATLAB프로그램 SobolIntegral101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: SobolIntegral101.m
%  Sobol 2-D Integration 1
%  Programmed by CBS
%-------------------------------------------------------------------------
function [xx MC QMC] = SobolIntegral101
format long
f = @(x,y) exp(-x.*y).*(sin(pi*x)+cos(pi*y));
Tint = dblquad(f,0,1,0,1)
Mmax = 20;
delta = 100;
rand('twister',5489)
psuedo = rand(Mmax*delta,2);
SobolSeq = sobolset(2)
Sseq = SobolSeq(1:Mmax*delta,:);
for m=1:Mmax
    Pran = f(psuedo(1:m*delta,1),psuedo(1:m*delta,2));
    MC(m) = mean(Pran);
    Qran = f(Sseq(1:m*delta,1),Sseq(1:m*delta,2));
    QMC(m) = mean(Qran);
end
% Plotting
xx = 1:Mmax;
plot([0 Mmax],[Tint Tint],'b-',xx,MC,'k-*', ...
      xx,QMC,'r--o','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xtick',0:5:20)
set(gca,'xticklabel',(0:5:20)*delta)
legend('Value','MC','Sobol-QMC','location','SE')
saveas(gcf,'SobolIntegral101','epsc')
save('SobolIntegral101','Sseq')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램은 다음 정적분을 몬테카를로법과 준몬테카를로법으로 계산하기 위한 것이다.

$$I \doteq \int_0^1 \int_0^1 \exp(-xy)\,[\sin \pi x + \cos \pi y]\,dxdy \tag{1}$$

예제 4.4.6에서와 마찬가지로 MATLAB함수 dblquad.m을 사용하면, 다음 식이 성립함을 알 수 있다.

$$I = 0.578396241990147 \tag{2}$$

이 MATLAB을 실행하면, 그림 4.6.2가 그려진다. 그림 4.6.2에서 흑색 실선은 몬테카를로적분을, 그리고 적색 긴점선은 Sobol점열을 사용한 준몬테카를로적분을 나타낸다. 이 그림에서 알 수 있듯이, 준몬테카를로적분이 몬테카를로적분보다 더 빨리 적분값에 수렴한다. 예제 4.4.6에서 Halton점열을 사용해서 계산한 준몬테카를로적분보다는 Sobol점열을 사용한 준몬테카를로적분이 약간 더 빨리 적분값에 수렴하는 것 같다. ■

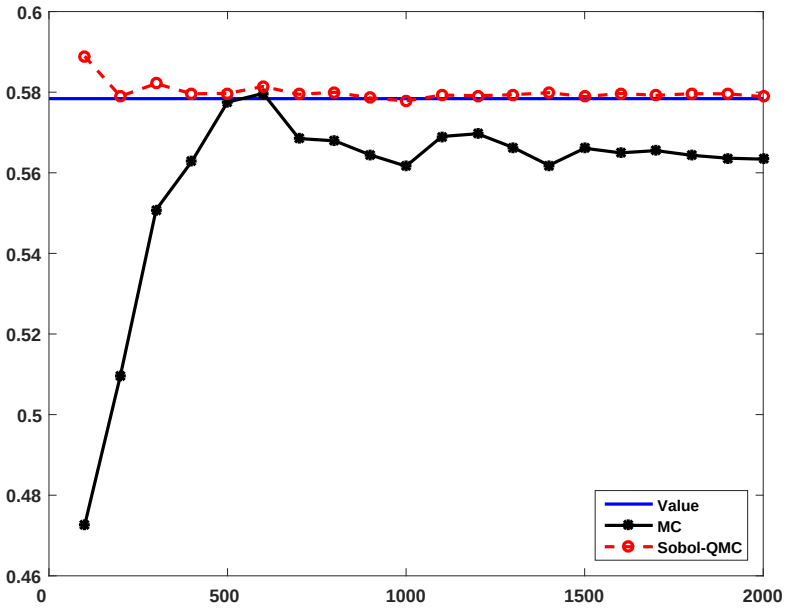


그림 4.6.2. Monte Carlo적분과 Sobol적분

예제 4.6.11 예제 4.4.6에서 Halton점열을 사용한 준몬테카를로적분, 예제 4.5.5에서 Faure 점열을 사용한 준몬테카를로적분, 그리고 예제 4.6.10에서 Sobol점열을 사용한 준몬테카를로 적분을 비교하기 위해서, 다음 MATLAB파일 QuasiMCIntegral101.m을 실행하라.

```
% ------------------------------------------------------------------------
%  Filename: QuasiMCIntegral101.m
%  Quasi-Monte Carlo Integrals 1
%  Programmed by CBS
%-------------------------------------------------------------------------
function QuasiMCIntegral101
format long
f = @(x,y) exp(-x.*y).*(sin(pi*x)+cos(pi*y));
Tint = dblquad(f,0,1,0,1)
Mmax = 20;
delta = 100;
[ MC HaltonInt ] = HaltonIntegral(f,Mmax,delta);
[ MC FaureInt ] = FaureIntegral(f,Mmax,delta);
[ MC SobolInt ] = SobolIntegral(f,Mmax,delta);
xx = 1:Mmax;
plot([0 Mmax],[Tint Tint],'c-',xx,HaltonInt,'b--', ...
      xx,FaureInt,'r-',xx,SobolInt,'k:','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xtick',0:5:20)
set(gca,'xticklabel',(0:5:20)*delta)
legend('Value','Halton-QMC','Faure-QMC','Sobol-QMC','location','NE')
saveas(gcf,'QuasiMCIntegral101','epsc')
save('QuasiMCIntegral101','HaltonInt','FaureInt','SobolInt')
end
% End of program
% ------------------------------------------------------------------------
function [ MC QMC] = HaltonIntegral(f,Mmax,delta);
%  Halton 2-D Integration
rand('twister',5489)
psuedo = rand(Mmax*delta,2);
```

```
vdc2 = VanDerCorput101(Mmax*delta,2);
vdc3 = VanDerCorput101(Mmax*delta,3);
for m=1:Mmax
    Pran = f(psuedo(1:m*delta,1),psuedo(1:m*delta,2));
    MC(m) = mean(Pran);
    Qran = f(vdc2(1:m*delta),vdc3(1:m*delta));
    QMC(m) = mean(Qran);
end
end
% ------------------------------------------------------------------------
function [ MC QMC] = FaureIntegral(f,Mmax,delta);
%  Faure 2-D Integration
rand('twister',5489)
psuedo = rand(Mmax*delta,2);
[Amat beta Fseq] = FaureSequence102(Mmax*delta,2);
for m=1:Mmax
    Pran = f(psuedo(1:m*delta,1),psuedo(1:m*delta,2));
    MC(m) = mean(Pran);
    Qran = f(Fseq(1:m*delta,1),Fseq(1:m*delta,2));
    QMC(m) = mean(Qran);
end
end
% ------------------------------------------------------------------------
function [ MC QMC] = SobolIntegral(f,Mmax,delta);
%  Sobol 2-D Integration
rand('twister',5489)
psuedo = rand(Mmax*delta,2);
SobolSeq = sobolset(2)
Sseq = SobolSeq(1:Mmax*delta,:);
for m=1:Mmax
    Pran = f(psuedo(1:m*delta,1),psuedo(1:m*delta,2));
    MC(m) = mean(Pran);
    Qran = f(Sseq(1:m*delta,1),Sseq(1:m*delta,2));
    QMC(m) = mean(Qran);
end
end
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 그림 4.6.3을 출력한다. 그림 4.6.3에서 알 수 있듯이, 차수가 낮은 Faure점열을 사용한 결과는 Sobol점열을 사용한 결과와 비슷하다. 또한, 이들이 Halton점열을 사용한 준몬테카를로적분보다 빨리 수렴한다고 할 수 있다. ■

제4.7절 준몬테카를로법에서의 정규확률분포

저불일치점열은 초입방체 $[0,1]^d$에서 일양확률분포하는 확률변수의 기대값을 계산하는데 사용되는 점열이다. 따라서, 확률변수들 사이에 상관이 있는 정규확률벡터에 준몬테카를로법을 적용하기 위해서, 이 정규확률벡터를 초입방체 $[0,1]^d$에서 일양확률분포하는 확률벡터로 변환한다.

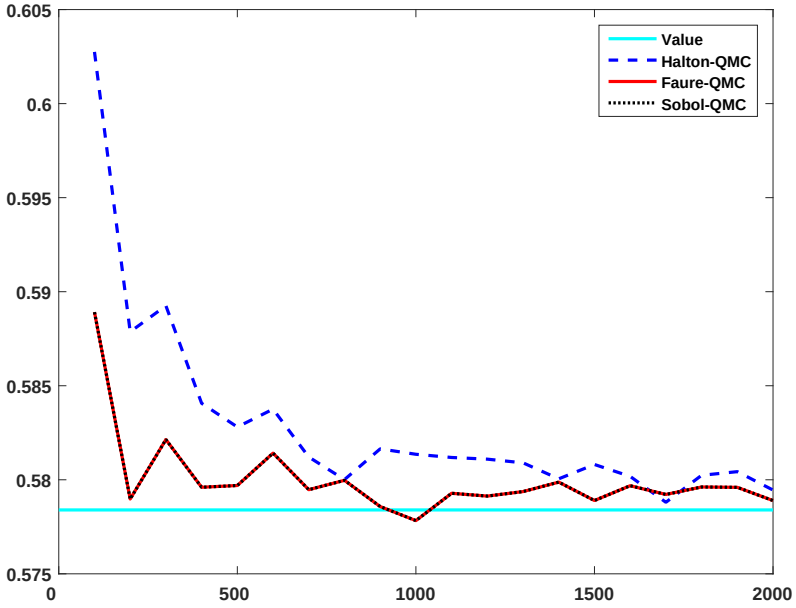


그림 4.6.3. 준몬테카를로적분

예제 4.4.8에서는 Box-Müller변환을 사용해서 2변량 정규확률분포에서 준난수들을 발생시켰다. 그러나, 이 방법을 적용하는데는 세심한 주의를 필요로 한다는 것을 알 수 있었다. 또한, 이 방법을 3변량 이상인 경우에 적용하는데는 더 큰 어려움이 따른다.

역함수법을 적용하면, 초입방체 $[0,1]^d$에서 일양확률분포하는 확률벡터 $\boldsymbol{x}$로부터 다변량정규확률분포를 따르며 분산공분산행렬이 단위행렬인 확률벡터 $\boldsymbol{y}$를 구할 수 있다. 또한, 확률벡터 $\boldsymbol{y}$에 Cholesky분해나 Schur분해를 적용해서 일반적인 다변량정규확률분포를 따르는 확률벡터 $\boldsymbol{z}$를 구할 수 있다. 따라서, 확률벡터 $\boldsymbol{z}$로부터 확률벡터 $\boldsymbol{x}$를 역추적하면, 일반적인 다변량정규확률분포를 따르는 확률벡터의 기대값 계산을 초입방체 $[0,1]^d$에서 일양확률분포를 따르는 확률벡터의 기대값 계산으로 치환할 수 있다. 즉, 일반적인 다변량정규확률분포를 다루는 문제가 준몬테카를로법으로 풀 수 있다. 그러나, 초입방체 $[0,1]^d$에서 일양확률분포를 따르는 확률벡터문제로 변환하는 것은 논리를 전개하기 위한 것이다. 실제 문제를 푸는 경우에는 준몬테카를로법을 사용할 수 있다는 것을 알고 있으므로, 보통 몬테카를로법과 마찬가지로 다음과 같은 알고리즘을 적용한다.

알고리즘 4.7.1: 다변량정규확률분포와 준몬테카를로법

(1단계) 저불일치점열 $\left\{\boldsymbol{u}_n = \left[u_n^{(1)}, u_n^{(2)}, \cdots, u_n^{(d)}\right]^t\right\}$를 생성한다.

(2단계) 저불일치점열 $\{\boldsymbol{u}_n\}$에 역함수법을 적용해서 d변량 표준정규확률분포를 따르는

점열 $\left\{\boldsymbol{x}_n = \left[x_n^{(1)}, x_n^{(2)}, \cdots, x_n^{(d)}\right]^t\right\}$을 만든다.

(3단계) 구하고자하는 점열들이 따르는 d차원 정규확률분포의 분산공분산행렬 Σ를 Cholesky분해하는 행렬 C를 구한다. 즉, 식 $\Sigma = C^tC$를 만족하는 행렬 C를 구한다. 또는 분산공분산행렬 Σ의 Schur분해 $\Sigma = Q\Lambda Q^t$를 구한다.

(4단계) 식 $\boldsymbol{y}_n = C^t\boldsymbol{x}_n$ 또는 식 $\boldsymbol{y}_n = Q\Lambda^{1/2}\boldsymbol{x}_n$을 만족하는 점열 $\left\{\boldsymbol{y}_n = \left[y_n^{(1)}, y_n^{(2)}, \cdots, y_n^{(d)}\right]^t\right\}$을 구한다.

(5단계) 주어진 다변량정규확률분포에 관한 문제를 점열 $\{\boldsymbol{y}_n\}$을 사용해서 준몬테카를로법으로 푼다.

예제 4.7.1 Sobol점열로부터 정규난수벡터들을 생성하기 위해서, 다음 MATLAB프로그램 SobolMultiNormalSequence101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: SobolMultiNormalSequence101.m
%  Multi-dimensional Sobol Sequence 101
%  Programmed by CBS
%-------------------------------------------------------------------------
clear all, close all, format short
diary SobolMultiNormalSequence101.txt
Nobs = 100000
% Generating sobol Sequence
SobolSeq = sobolset(3);
Sseq = SobolSeq(1000+1:1000+Nobs,:);
    % Correlation of Quasi-Random Numbers
    corrQR = corr(Sseq)
% Generating Standard Multivariate Normal Random Vectors
X = norminv(Sseq);
% Covariance of Standard Multi-Normal Random Numbers
    meanSMN = mean(X)
    covSMN = cov(X)
% Given Variance-Covariance Matrix
A = [ 1 0 0 ; -0.2 1 0 ; 0.3 -0.4 1 ];
Sigma = A*A'
% Cholesky Decomposition
C = chol(Sigma)
    % DoubleCheck = Sigma - C'*C
Y1 = X*C;
    % Covariance of Multi-Normal Random Numbers
    meanMNc = mean(Y1)
    covMNc = cov(Y1)
% Schur Decomposition
[Q Lambda] = eig(Sigma)
    % DoubleCheck = Sigma - Q*Lambda*Q'
Y2 = X*Q*sqrt(Lambda)*Q';
    % Covariance of Multi-Normal Random Numbers
    meanMNs = mean(Y2)
    covMNs = cov(Y2)
diary off
% End of program
```

```
38 % ------------------------------------------------------------------------
```

이 MATLAB프로그램은 다음과 같은 다변량정규확률분포에서 준난수벡터들을 생성하기 위한 것이다.

$$\boldsymbol{y} \overset{d}{\sim} N(\mathbf{0}, \Sigma) \tag{1}$$

여기서 분산공분산행렬 Σ는 다음과 같다.

$$\Sigma = \begin{bmatrix} 1.0000 & -0.2000 & 0.3000 \\ -0.2000 & 1.0400 & -0.4600 \\ 0.3000 & -0.4600 & 1.2500 \end{bmatrix} \tag{2}$$

MATLAB함수들 sobolset.m을 사용해서, 3차원 Sobol점열들을 생성하였다. 생성된 첫 번째 Sobol점은 $[0,0,0]$이고, 역함수법을 적용해서 이 점으로부터 정규난수벡터를 생성하는데 문제가 있다. 따라서, 첫 1000개 점들은 버리고, 그 다음 10만개 Sobol점들을 사용하기로 하자. 이 Sobol점들을 구성하는 행렬 Sseq가 서로 독립인 행들로 이루졌는지를 확인하기 위해서 이들의 상관계수행렬을 구하면, 그 결과는 다음과 같다.

$$corr_{QR} = \begin{bmatrix} 1.0000 & 0.0000 & 0.0000 \\ 0.0000 & 1.0000 & -0.0000 \\ 0.0000 & -0.0000 & 1.0000 \end{bmatrix} \tag{3}$$

이 상관계수행렬은 단위행렬에 가깝다. 즉, 행렬 Sseq가 서로 독립인 행들로 구성되어있다고 할 수 있다.

MATLAB함수 norminv.m을 사용해서, 이 10만개 Sobol점들을 정규난수벡터들 $\{\boldsymbol{x}_n\}$으로 변환한 것이 행렬 X이다. 이 행렬의 평균벡터 $\bar{\boldsymbol{x}}$와 분산공분산추정행렬 $\hat{\Sigma}_x$는 각각 다음과 같다.

$$\bar{\boldsymbol{x}} = \frac{1}{10^4} \begin{bmatrix} -0.9141 \\ -0.1225 \\ 0.7238 \end{bmatrix}, \quad \hat{\Sigma}_x = \begin{bmatrix} 1.0000 & 0.0000 & -0.0001 \\ 0.0000 & 1.0000 & -0.0001 \\ -0.0001 & -0.0001 & 0.9999 \end{bmatrix} \tag{4}$$

Cholesky분해를 사용해서 식 (1)의 다변량정규확률분포로부터 준난수벡터들 생성하기로 하자. MATLAB함수 chol.m을 사용해서 분산공분산추정행렬 $\hat{\Sigma}_x$를 Cholesky분해한 결과는

다음과 같다.

$$\hat{\Sigma}_x = C^t C, \quad C = \begin{bmatrix} 1.0000 & -0.2000 & 0.3000 \\ 0 & 1.0000 & -0.4000 \\ 0 & 0 & 1.0000 \end{bmatrix} \tag{5}$$

벡터들 $\{\boldsymbol{y}_n^{(1)} \doteq C^t \boldsymbol{x}_n\}$은 식 (1)의 다변량정규확률분포에서 발생한 준난수벡터들이다. 이들의 평균벡터 $\bar{\boldsymbol{y}}^{(1)}$와 분산공분산추정행렬 $\hat{\Sigma}_y^{(1)}$는 각각 다음과 같다.

$$\bar{\boldsymbol{y}}^{(1)} = \frac{1}{10^4} \begin{bmatrix} -0.9141 \\ 0.0603 \\ 0.4986 \end{bmatrix}, \quad \hat{\Sigma}_y^{(1)} = \begin{bmatrix} 1.0000 & -0.2000 & 0.2999 \\ -0.2000 & 1.0400 & -0.4600 \\ 0.2999 & -0.4600 & 1.2499 \end{bmatrix} \tag{6}$$

식 (6)의 분산공분산행렬 $\hat{\Sigma}_y^{(1)}$은 식 (2)의 분산공분산행렬 Σ와 비슷하다.

Schur분해를 사용해서 식 (1)의 다변량정규확률분포로부터 준난수벡터들 생성하기로 하자. MATLAB함수 eig.m을 사용해서 분산공분산행렬 Σ를 Schur분해한 결과는 다음과 같다.

$$\Sigma = Q \Lambda Q^t \tag{7}$$

$$Q = \begin{bmatrix} -0.1547 & 0.8953 & 0.4178 \\ 0.7345 & 0.3870 & -0.5574 \\ 0.6607 & -0.2206 & 0.7175 \end{bmatrix}, \quad \Lambda Q = \begin{bmatrix} 0.6684 & 0 & 0 \\ 0 & 0.8396 & 0 \\ 0 & 0 & 1.7820 \end{bmatrix} \tag{8}$$

벡터들 $\{\boldsymbol{y}_n^{(2)} \doteq Q \Lambda^{1/2} Q^t \boldsymbol{x}_n\}$은 식 (2)의 다변량정규확률분포에서 발생한 준난수벡터들이다. 이들의 평균벡터 $\bar{\boldsymbol{y}}^{(2)}$와 분산공분산추정행렬 $\hat{\Sigma}_y^{(2)}$는 각각 다음과 같다.

$$\bar{\boldsymbol{y}^{(2)}} = \frac{1}{10^4} \begin{bmatrix} -0.7935 \\ -0.1987 \\ 0.6904 \end{bmatrix}, \quad \hat{\Sigma}_y^{(2)} = \begin{bmatrix} 1.0000 & -0.2000 & 0.2999 \\ -0.2000 & 1.0400 & -0.4600 \\ 0.2999 & -0.4600 & 1.2499 \end{bmatrix} \tag{9}$$

식 (9)의 분산공분산행렬 $\hat{\Sigma}_y^{(2)}$는 식 (6)의 분산공분산행렬 $\hat{\Sigma}_y^{(1)}$과 동일하다. ■

제4.8절 준몬테카를로법과 금융파생상품의 가치평가

다양한 종류의 준난수들이 존재하고, 이들을 비교하는 문헌들도 많이 있다. 어떤 문헌에서는 특정 준난수가 다른 것보다 더 우수하다고 주장하고 있지만, 그러한 결과는 넓은 범위에서 한 실험을 바탕으로 하는 것이 아닌 경우가 많다. 다음과 같은 견해가 일반적으로 받아들여지고 있다. 적당한 차원, 예를 들어 15이하인 경우에는 Halton점열, Faure점열 그리고 Sobol점열 사이에 큰 차이는 없다. Sobol점열은 초기값에 따라 수렴속도가 다르고, Faure점열보다 Sobol 점열을 빠르게 생성할 수 있다. 일반적으로 초입방체 $[0,1)^d$에서 준난수열이 의사난수열보다 더 균등하게 분포하는 것이 일반적이다. 따라서, 저차원 또는 중간정도의 차원의 문제에서는 몬테카를로법보다 준몬테카를로법이 더 좋은 추정값을 제공한다. 또한, 준난수열의 생성속도가 의사난수열보다 빠르다. 그러나, 기수를 어떻게 선택하느냐에 따라 준난수열을 구성하는 점들의 상관관계가 클 수도 있다. 이렇게 상관관계가 큰 경우에는 발생된 점열들이 어떤 초평면(hyperplane) 주변에 군집될 것이고, 따라서 계산하고자 하는 정적분값은 이 초평면 주변에 큰 가중값을 주게 될 것이고, 결과적으로 편의(bias)를 갖게 될 것이다. 오차의 차수라는 관점에서보면, 1차원인 경우에는 몬테카를로법이나 준몬테카를로법보다 사다리꼴공식을 이용하는 것이 효율적이다.

컬럼비아대학교 컴퓨터과학과 교수인 J. F. Traub는 1994년 그의 박사과정학생이던 S. Paskoc에게 골드만삭스로부터 받은 CMO(collateralized mortgage obligation)의 공정한 가치를 몬테카를로법과 준몬테카를로법을 사용해서 계산하고 비교하게 하였다. 이 문제는 차원이 $k = 360$인 적분문제를 수치적으로 구하는 것이었다. Paskoc은 Sobol점열을 사용한 준몬테카를로법이 의사난수를 사용한 몬테카를로법에 대해, 항상 우위에 있다는 결론을 내렸다. 이는 상당히 놀라운 결과였다. 재무학이나 금융공학에서는 이러한 문제를 풀기 위해서 몬테카를로법을 주로 사용해왔고, 정수론(number theory)을 전공하는 학자들은 차수가 $k = 12$보다 큰 경우에는 준몬테카를로법을 사용해서는 안된다고 믿어왔다. Paskoc와 Traub은 그들의 결과를 월스트리트의 회사들에게 알려주었고, 1995년 Journal of Portfolio Management에 발표하였다. Traub과 제자들은 FinDer Software System이라는 회사를 차렸고, 특이하게도 준몬테카를로법을 바탕으로 U.S. patents US5940810와 US0605837를 획득했다. 이 특허권은 금융파생상품의 가치를 계산하는 잘 알려진 문제에 잘 알려진 기법인 준난수를 적용한 것이다. 본저자는 특허권이 남용된 것이 아닌가 생각한다.

Paskov & Traub [40] 이후, 재무학에서 발생하는 고차원 문제를 푸는데 일반적으로 준몬테카를로법이 몬테카를로법보다 우수하다는 결론을 내린 논문으로는 Caflisch & Morokoff

& Owen [13], Joy & Boyle & Tan [22], Ninomiya & Tezuka [38], Papageorgiou & Traub [39], Ackworth & Broadie & Glasserman [5] 등이 있다. 특히, Ninomiya & Tezuka [38]는 전통적인 저불일치점열인 Halton점열, Sobol점열, Faure점열과 새로운 저불일치점열인 일반화Niederreiter점열을 사용해서, MBS(mortgage-backed securities)의 가치와 이색옵션의 가치를 평가했다. 그들은 일반화Niederreiter점열이 전통적인 저불일치점열이나 의사난수열보다 수렴이 빠르고, 전통적인 저불일치점열은 의사난수열보다 좋지 않은 경우가 있음을 보였다. 또한, Papageorgiou & Traub [39]은 Sobol점열과 일반화Faure점열을 사용해서, CMO가치를 평가했다. 그들은 저불일치점열들 모두가 의사난수열에 비해 수렴이 빠르고, 특히 일반화Faure점열이 우수하다고 보고하고 있다. 결론적으로 말하면, 일반적으로 높은 차원의 정적분을 계산하는데는 몬테카를로법보다 준몬테카를로법이 유용하다고 알려져 있다. 그러나, 일반적으로 준난수열은 난수성검정을 통과하지 못한다. 카이제곱검정이나 Kolmogorov-Smirnov검정과 같은 전통적인 난수성검정을 통과하기에는 준난수들은 지나치게 균등하다. 의사난수열과는 달리 준난수열은 난수성을 추구하는 것이 아니라 균등성 즉 일양성을 추구하기때문에, 이러한 현상이 나타나는 것은 당연하다. 이러한 준난수열의 결정성(deterministic property)때문에 준몬테카를로법을 통해 얻은 결과에 통계적 기법을 적용하지 않는다. 더구나 준몬테카를로법에 의한 오차를 해석적으로 분석하는 것은 어려운 일이다. 오늘날 금융파생상품의 가치를 평가하는데 준몬테카를로법이 널리 사용되고 있다. 그렇다고, 준몬테카를로법이 만병통치약이라는 것은 아니다. 오늘날에도 몬테카를로법과 준몬테카를로법을 비교하는 연구가 계속되고 있고, 특히 몬테카를로법보다 준몬테카를로법이 유용한 경우에 그렇게 되는 이유가 무엇인지에 대한 연구가 활발하게 진행되고 있다.

제 5 장

확률과정 생성

이 장에서는 몬테카를로 시뮬레이션에 사용되는 중요한 확률과정들에서 표본경로를 생성하는 방법을 살펴보자. 이 장의 원고를 작성하는데 주로 Kroese & Taimre & Botev [26]와 MATLAB Financial Toolbox를 참고하였다. 이 툴박스에 대해서는 다음 웹사이트를 참조하라.

http://kr.mathworks.com/help/finance/index.html?s_cid=doc_ftr

이 장에서는 확률과정 $\{x_t\}$를 $\{x(t)\}$로도 표기할 것이다. 비록 LaTeX을 사용한다해도, 첨자의 첨자의 첨자를 표기하면 활자가 너무 작아지기 때문이다.

제5.1절 Gauss-Markov확률과정

Markov성을 갖는 Gauss과정을 Gauss-Markov확률과정이라 한다. Gauss-Markov확률과정 $\{x_t \mid t = 1, 2, \cdots, n\}$가 다음 식들을 만족한다고 하자.

$$\begin{bmatrix} x_t \\ x_{t+1} \end{bmatrix} \overset{d}{\sim} \mathcal{N}\left(\begin{bmatrix} \mu_t \\ \mu_{t+1} \end{bmatrix}, \begin{bmatrix} \sigma_{t,t} & \sigma_{t,t+1} \\ \sigma_{t+1,t} & \sigma_{t+1,t+1} \end{bmatrix} \right), \quad (t = 1, 2, \cdots, n-1) \tag{5.1.1}$$

여기서 $\sigma_{t,t+1} = \sigma_{t+1,t}$임에 유의하라. 각 $t(= 1, 2, \cdots, n-1)$에 대해서 다음 식이 성립한다.

$$x_{t+1} \mid x_t \overset{d}{\sim} \mathcal{N}\left(\mu_{t+1} + \frac{\sigma_{t+1,t}}{\sigma_{t,t}} \left[x_t - \mu_t \right], \sigma_{t+1,t+1} - \frac{\sigma_{t+1,t}^2}{\sigma_{t,t}} \right) \tag{5.1.2}$$

식 (5.1.2)를 다음과 같이 쓸 수 있다.

$$x_{t+1} = \mu_{t+1} + \frac{\sigma_{t+1,t}}{\sigma_{t,t}}\left[x_t - \mu_t\right] + \epsilon_{t+1} \tag{5.1.3}$$

$$\epsilon_{t+1} \overset{d}{\sim} \mathcal{N}\left(0, \sigma_{t+1,t+1} - \frac{\sigma_{t+1,t}^2}{\sigma_{t,t}}\right) \tag{5.1.4}$$

식 (5.1.4)의 ϵ_{t+1}은 오차항에 해당하는 것이다. 따라서, 다음 알고리즘을 사용해서 Gauss-Markov확률과정에서 표본경로를 발생시킬 수 있다.

알고리즘 5.1.1: Gauss-Markov확률과정

(1단계) 표준정규난수 z_1을 사용해서, $x_1 = \mu_1 + \sqrt{\sigma_{1,1}} z_1$을 계산한다.

(2단계) 표준정규난수열 $\{z_t \mid t = 2, 3, \cdots, n\}$을 발생시킨 뒤, 다음 값들을 계산한다.

$$x_{t+1} = \mu_{t+1} + \frac{\sigma_{t+1,t}}{\sigma_{t,t}}\left[x_t - \mu_t\right] + \sqrt{\sigma_{t+1,t+1} - \frac{\sigma_{t+1,t}^2}{\sigma_{t,t}}} z_{t+1}, \quad (t = 1, 2, \cdots, n-1)$$

예제 5.1.1 알고리즘 5.1.1을 사용해서 다음과 같은 Gauss-Markov확률과정의 표본경로 $\{x_t\}$를 발생시키기로 하자.

$$\begin{bmatrix} x_t \\ x_{t+1} \end{bmatrix} \overset{d}{\sim} \mathcal{N}\left(\begin{bmatrix} t \\ t+1 \end{bmatrix}, \begin{bmatrix} t & 0.3t \\ 0.3t & t+1 \end{bmatrix}\right), \quad (t = 1, 2, \cdots, n-1) \tag{1}$$

이러한 표본경로를 발생시키기 위해서, 다음 MATLAB프로그램 GaussMarkovProcess1D101.m을 실행해보자.

```
% ------------------------------------------------------------------------
%  Filename: GaussMarkovProcess1D101.m
%  Generating 1-D Gauss Markov Stochastic Process
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all, clc
Ns = 60;
rand('twister',5489)
x = zeros(Ns,1); tt = (1:Ns)';
rho = 0.3
mu = @(t) t
sigma11 = @(t) t
sigma12 = @(t) rho*t
for jj = 1:Ns-1
    x(jj+1) = mu(jj+1)+sigma12(jj)/sigma11(jj)*(x(jj)-mu(jj)) ...
```

```
            + sqrt(sigma11(jj+1)-sigma12(jj)^2/sigma11(jj)) ...
            *randn(1);
end
% Plotting
plot(tt,x,'k-','linewidth',2);
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf t')
ylabel('\bf x_{t}','rotation',0)
saveas(gcf,'GaussMarkovProcess1D101','epsc')
save('GaussMarkovProcess1D101','x')
% End of program
% --------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 식 (1)을 만족하는 Gauss-Markov확률과정의 표준 Brown운동의 표본경로 $\{x(t_j)\}$를 출력한다. 이 표본경로가 그림 5.1.1에 그려져 있다. ■

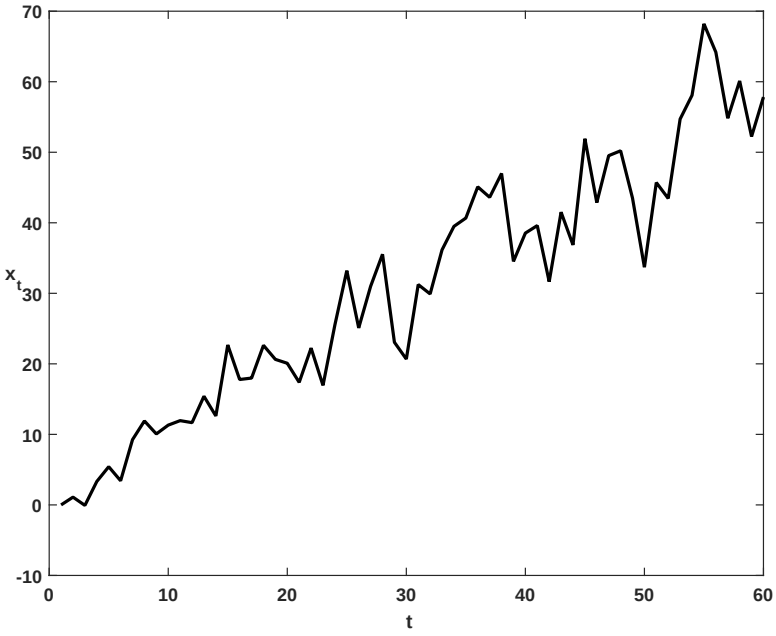


그림 5.1.1. Gauss-Markov확률과정의 표본경로

만약 $\mu_{t+1}, \sigma_{t,t}$ 그리고 $\sigma_{t+1,t}$가 시점 t에 의존하지 않는 상수라면, 식 (5.1.3)과 식 (5.1.4)를 다음과 같이 쓸 수 있다.

$$x_{t+1} = \mu + \phi\,[x_t - \mu] + \epsilon_{t+1} \tag{5.1.5}$$

여기서 $\{\epsilon_t\}$는 서로 독립이며 동일한 정규확률분포를 따르는 확률과정이다. 식 (5.1.5)를 만족하는 확률과정을 1차 자기회귀과정 (autoregressive process of order 1)이라 부른다.

알고리즘 5.1.1을 다변량Gauss과정(multidimensional Gasuss-Markov stochastic pro-

cess) 으로 확장할 수 있다. 확률벡터과정 $\{\boldsymbol{x}_t \mid t = 1, 2, \cdots, n\}$ 가 다음 식들을 만족한다고 하자.

$$\begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{x}_{t+1} \end{bmatrix} \overset{d}{\sim} \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_t \\ \boldsymbol{\mu}_{t+1} \end{bmatrix}, \begin{bmatrix} \Sigma_{t,t} & \Sigma_{t,t+1} \\ \Sigma_{t+1,t} & \Sigma_{t+1,t+1} \end{bmatrix} \right), \quad (t = 1, 2, \cdots, n-1) \tag{5.1.6}$$

여기서 $\Sigma_{t,t+1}^t = \Sigma_{t+1,t}$ 임에 유의하라. 각 $t(= 1, 2, \cdots, n-1)$ 에 대해서 다음 식이 성립한다.

$$\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t \overset{d}{\sim} \mathcal{N}\left(\boldsymbol{\mu}_{t+1} + \Sigma_{t+1,t}\Sigma_{t,t}^{-1}\left[\boldsymbol{x}_t - \boldsymbol{\mu}_t\right], \Sigma_{t+1,t+1} - \Sigma_{t+1,t}\Sigma_{t,t}^{-1}\Sigma_{t,t+1}\right) \tag{5.1.7}$$

여기서 $\Sigma_{t+1,t+1\cdot t} \doteq \Sigma_{t+1,t+1} - \Sigma_{t+1,t}\Sigma_{t,t}^{-1}\Sigma_{t,t+1}$ 이다. 따라서, 다음 알고리즘을 사용해서 다변량 Gauss-Markov확률과정으로부터 표본경로를 발생시킬 수 있다.

알고리즘 5.1.2: 다변량 Gauss-Markov확률과정

(1단계) 행렬 $\Sigma_{1,1}$ 을 Cholesky분해한 하삼각행렬 L_1 을 구한다. 즉, 식 $\Sigma_{1,1} = L_1 L_1^t$ 을 만족하는 행렬 L_1 을 구한다. 표준정규난수벡터 $\boldsymbol{z}_1$ 을 사용해서, 다음과 같이 벡터 $\boldsymbol{x}_1$ 을 발생시킨다.

$$\boldsymbol{x}_1 = \boldsymbol{\mu}_1 + L_1 \boldsymbol{z}_1$$

(2단계) 다음 행렬 $\Sigma_{t+1,t+1\cdot t}$ 를 Cholesty분해한 하삼각행렬 L_{t+1} 을 구한다. 즉, 다음 식을 만족하는 행렬 L_{t+1} 을 구한다.

$$\Sigma_{t+1,t+1\cdot t} = L_{t+1} L_{t+1}^t$$

표준정규난수벡터열 $\{\boldsymbol{z}_t \mid t = 2, 3, \cdots, n\}$ 을 발생시킨 뒤, 다음 벡터들을 계산한다.

$$\boldsymbol{x}_{t+1} = \boldsymbol{\mu}_{t+1} + \Sigma_{t+1,t}\Sigma_{t,t}^{-1}\left[\boldsymbol{x}_t - \boldsymbol{\mu}_t\right] + L_{t+1} z_{t+1}, \quad (t = 1, 2, \cdots, n-1)$$

예제 5.1.2 알고리즘 5.1.2를 사용해서 다음과 같은 2변량 Gauss-Markov 확률과정으로부터 표본경로 $\{\boldsymbol{x}_t\}$ 를 발생시키기로 하자.

$$\begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{x}_{t+1} \end{bmatrix} \overset{d}{\sim} \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_t \\ \boldsymbol{\mu}_{t+1} \end{bmatrix}, \begin{bmatrix} \Sigma_{t,t} & \Sigma_{t,t+1} \\ \Sigma_{t+1,t} & \Sigma_{t+1,t+1} \end{bmatrix} \right), \quad (t = 1, 2, \cdots, n-1) \tag{1}$$

여기서 평균벡터와 분산공분산행렬들은 다음과 같다.

$$\boldsymbol{\mu}_t = \begin{bmatrix} t \\ t+1 \end{bmatrix}, \quad \Sigma_{t,t} = \begin{bmatrix} t & \frac{t}{2} \\ \frac{t}{2} & t \end{bmatrix}, \quad \Sigma_{t,t+1} = \begin{bmatrix} \frac{t}{4} & \frac{t}{8} \\ \frac{t}{8} & \frac{t}{2} \end{bmatrix} \tag{2}$$

이러한 표본경로를 발생시키기 위해서, 다음 MATLAB프로그램 GaussMarkovProcess2D1.m을 실행해보자.

```
% ------------------------------------------------------------------------
%  Filename: GaussMarkovProcess2D101.m
%  Generating 2-D Gauss Markov Stochastic Process
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all, clc
Ns = 60;
rand('twister',5489)
x = zeros(2,Ns); tt = (1:Ns);
mu = @(t) [ t -t ]'
Sigma11 = @(t) [ t t/2 ; t/2 t ]
Sigma12 = @(t) [ t/4 t/8 ; t/8 t/4 ]
for jj = 1:Ns-1
    dum =  Sigma11(jj+1)-Sigma12(jj)'*Sigma11(jj)^(-1)*Sigma12(jj);
    x(:,jj+1) = mu(jj+1)+Sigma12(jj)*Sigma11(jj)^(-1) ...
                *(x(:,jj)-mu(jj))+ sqrt(dum)*randn(2,1);
end
% Plotting
plot(x(1,:),x(2,:),'k-','linewidth',2);
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf x_{1,t}')
ylabel('\bf x_{2,t}','rotation',0)
axis square
saveas(gcf,'GaussMarkovProcess2D101','epsc')
save('GaussMarkovProcess2D101','x')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 식 (1)과 식 (2)를 만족하는 2변량 Gauss-Markov 확률과정의 표본경로 $\{\boldsymbol{x}_t\}$를 출력한다. 이 표본경로의 산점도가 그림 5.1.2에 그려져 있다. ■

제5.2절 Brown운동류의 생성

금융공학에서 Brown운동류는 매우 중요한 역할을 한다. Brown운동을 다음과 같이 정의한다.

정의 5.2.1

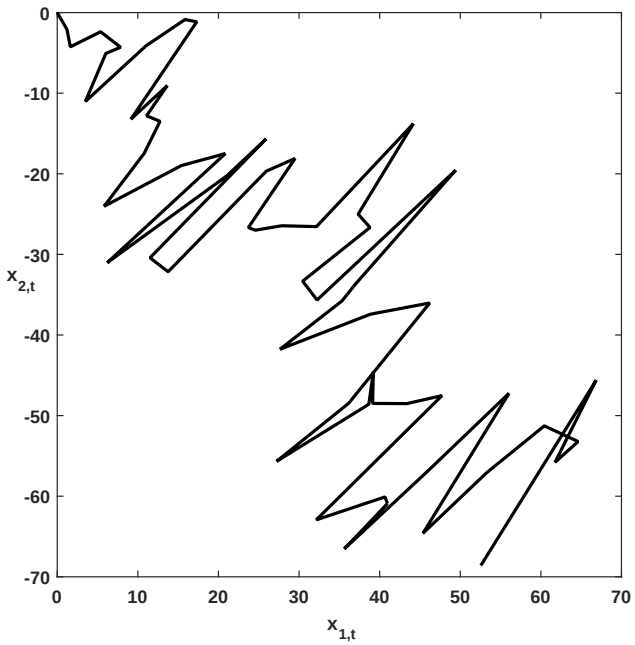


그림 5.1.2. 이변량 Gauss-Markov확률과정의 표본경로

확률과정 $\{W_t \mid t \geq 0\}$가 다음 조건들을 만족하면, $\{W_t\}$를 확산모수(diffusion parameter)가 σ인 Brown운동 또는 Wiener과정이라 한다.

a) 각 $t(\geq 0)$에 대해서, 확률변수 W_t는 $\mathcal{N}(0, \sigma^2 t)$를 따른다.

b) 확률과정 $\{W_t\}$에서 실현된 표본경로는 t의 연속함수이다.

만약 확산모수가 $\sigma = 1$이면, $\{W_t\}$를 표준Brown운동이라 부른다. Brown운동의 표본경로를 다음과 같이 생성할 수 있다.

알고리즘 5.2.1: Brown운동 생성

표준정규난수열 $\{z_j \mid j = 1, 2, \cdots, n\}$을 발생시킨 뒤, 다음 값들을 계산한다.

$$W_{t_j} = \sum_{k=1}^{j} \sqrt{t_k - t_{k-1}} z_k, \quad (j = 1, 2, \cdots, n-1)$$

여기서 $0 = t_0 < t_1 < \cdots < t_n$이다. 이 $\left\{W_{t_j} \mid j = 1, 2, \cdots, n\right\}$가 Brown운동의 표본경로이다.

Brown운동의 표본경로는 연속이다. 따라서, 소구간 $[t_{j-1}, t_j]$에서 점 $\left(t_{j-1}, W_{t_{j-1}}\right)$ 과 점 $\left(t_j, W_{t_j}\right)$를 연결할 필요가 있다. 선형보간을 적용하면, 각 $s\,(\in (t_{j-1}, t_j))$에 대해서 다음

근사값을 구할 수 있다.

$$W_s = \frac{[t_j - s]\, W_{t_{j-1}} + [s - t_{j-1}]\, W_{t_j}}{t_j - t_{j-1}} \tag{5.2.1}$$

식 (5.2.1)은 선형보간을 사용해서 구한 W_s의 근사값이다. 물론, 스플라인과 같은 다른 보간법을 사용하거나, 뒤에서 다루는 Brown다리의 생성법을 이용할 수 있다. 그러나, $\max_j \mid t_j - t_{j-1} \mid$ 를 작게 하는 것이 Brown운동의 표본경로를 생성하는 좋은 방법이라고 생각한다.

예제 5.2.1 알고리즘 5.2.1을 사용해서 표준Brown운동의 표본경로를 발생시키기 위해서, 다음 MATLAB프로그램 BrownianMotion1D101.m을 실행해보자.

```
% ------------------------------------------------------------------------
%  Filename: BrownianMotion1D101.m
%  Generating 1-D Standard Brownian Motion for T seconds
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all, clc
T = 3;                              % Time in Seconds
Deltat = 1/120;
N = T/Deltat;                       % Number of Samples
tt = Deltat*(1:N)';
sqDeltat = sqrt(Deltat);
rand('twister',5489)
W = zeros(N,1);
for ii = 1:N-1
    W(ii+1) = W(ii) + sqDeltat*randn(1);
end
% Plotting
plot(tt,W,'k-','linewidth',2);
set(gca,'fontsize',11,'fontweigh','bold')
hold on
plot([0 T],[0 0],'r-','linewidth',1.5);
xlabel('\bf t')
ylabel('\bf W_{t}','rotation',0)
hold off
saveas(gcf,'BrownianMotion1D101','epsc')
save('BrownianMotion1D101','W')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 표준Brown운동의 표본경로를 출력한다. 이 표본경로가 그림 5.2.1에 그려져 있다. ■

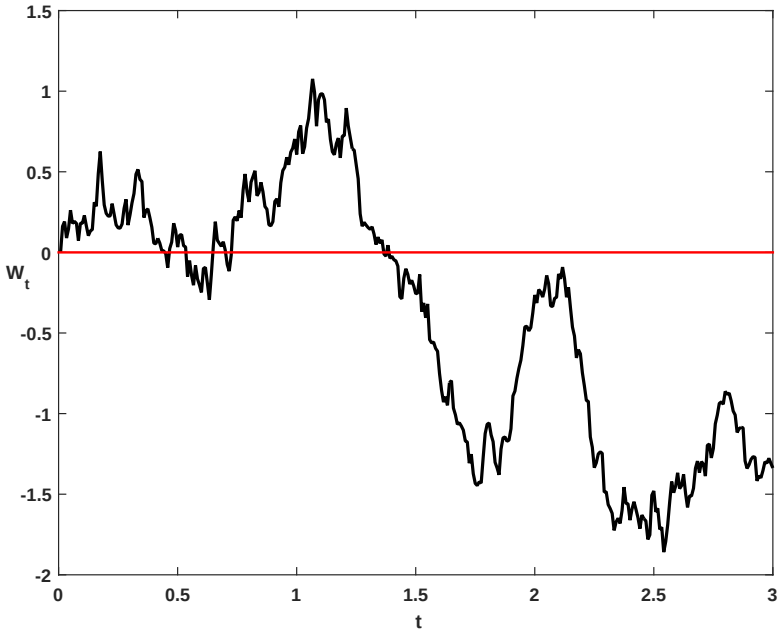


그림 5.2.1. 표준Brown운동의 표본경로

정의 5.2.2: 추세Brown운동

확률과정 $\{x_t \mid t \geq 0\}$가 다음 조건들을 만족하면, 추세Brown운동 또는 추세Wiener 과정이라 한다.

a) 각 $t(\geq 0)$에 대해서, 확률변수 x_t는 $\mathcal{N}(\mu t, \sigma^2 t)$를 따른다. 이 경우에 μ를 추세모수(drift parameter) 그리고 σ를 확산모수라 한다.

b) 확률과정 $\{x_t\}$에서 실현된 표본경로는 t의 연속함수이다.

정의 5.2.2에서 알 수 있듯이, 추세모수가 μ이고 확산모수 σ인 추세Brown운동 $\{x_t \mid t \geq 0\}$를 다음과 같이 나타낼 수 있다.

$$x_t = \mu t + \sigma W_t \tag{5.2.2}$$

여기서 $\{W_t \mid t \geq 0\}$는 표준Brown운동이다.

알고리즘 5.2.2: 추세Brown운동

(1단계) 표준Brown운동의 표본경로 $\{W_{t_j} \mid j = 0, 1, 2, \cdots, n\}$를 구한다. 여기서 $W_{t_0} = W_0 = 0$이다.

(2단계) 수열 $\{x_{t_j} = \mu t_j + \sigma W_{t_j} \mid j = 0, 1, 2, \cdots, n\}$를 계산한다.

이 $\{x_{t_j} \mid j = 1, 2, \cdots, n\}$가 추세Brown운동의 표본경로이다.

예제 5.2.2 알고리즘 5.2.2를 사용해서 추세Brown운동의 표본경로를 발생시키기 위해서, 다음 MATLAB프로그램 DriftBrownianMotion1D101.m을 실행해보자.

```
% ------------------------------------------------------------------------
%  Filename: DriftBrownianMotion1D101.m
%  Generating 1-D Brownian Motion with a Drift
%  Programmed by CBS
%-------------------------------------------------------------------------
clear all, close all, clc
T = 3;                              % Time in Seconds
Deltat = 1/120;
N = T/Deltat;                       % Number of Samples
tt = Deltat*(0:N-1)';
rand('twister',5489)
W = [0; cumsum(randn(N-1,1))]*sqrt(Deltat);
mu = 1                              % Drift parameter
sigma = 0.70                        % Diffusion parameter
DriftB = mu*tt + sigma*W;
% Plotting
plot(tt,DriftB,'k-','linewidth',2);
set(gca,'fontsize',11,'fontweigh','bold')
hold on
plot([0 T],[0 0],'r-','linewidth',1.5);
xlabel('\bf t')
ylabel('\bf B_{t}','rotation',0)
hold off
saveas(gcf,'DriftBrownianMotion1D101','epsc')
save('DriftBrownianMotion1D101','W')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 추세Brown운동의 표본경로 $\{x_{t_j}\}$를 출력한다. 이 표본경로가 그림 5.2.2에 그려져 있다. ■

정의 5.2.3: 기하Brown운동

추세모수 μ 와 확산모수 σ를 갖는 추세Brown운동 $\{x_t \mid t \geq 0\}$에 대해서, $\left\{x_t = \exp\left(\left[\mu - \frac{1}{2}\sigma^2\right] t + \sigma W_t\right) \mid t \geq 0\right\}$를 추세모수 μ 와 확산모수 σ를 갖는 기하Brown운동 또는 기하Wiener과정이라 한다.

정의 5.2.3와는 달리, 문헌에 따라서는 표준Brown운동 $\{W_t \mid t \geq 0\}$에 대해서

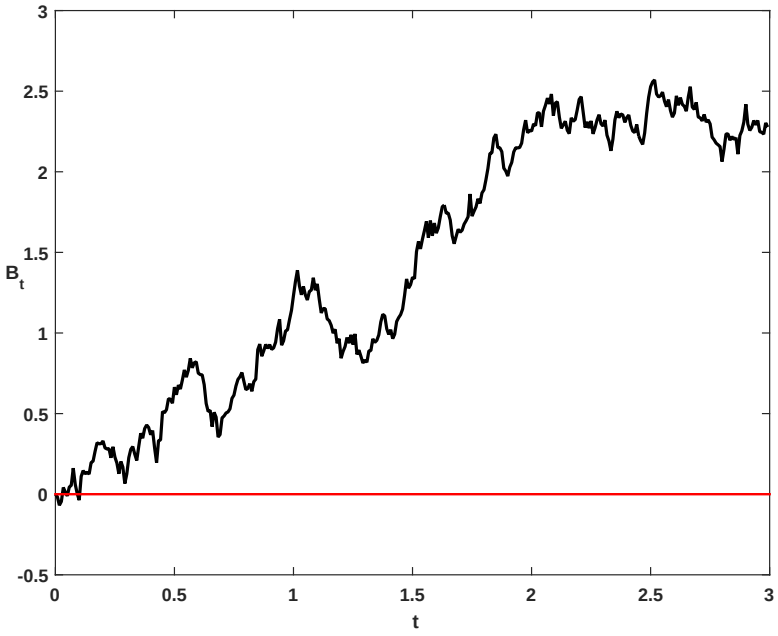


그림 5.2.2. 추세Brown운동의 표본경로

$\{x_t = \exp(\mu t + \sigma W_t) \mid t \geq 0\}$를 추세모수 μ 와 확산모수 σ를 갖는 기하Brown운동이라고 하기도 한다. 정의 5.2.3의 추세Brown운동의 평균과 분산은 각각 다음과 같다.

$$E(x_t) = \exp(\mu t), \quad Var(x_t) = \exp(2\mu t)\left[\exp\left(\sigma^2 t\right) - 1\right] \tag{5.2.3}$$

알고리즘 5.2.3: 기하Brown운동

(1단계) 표준Brown운동의 표본경로 $\{W_{t_j} \mid j = 0, 1, 2, \cdots, n\}$를 구한다. 여기서 $W_{t_0} = W_0 = 0$이다.

(2단계) 수열 $\left\{x_{t_j} = \exp\left(\left[\mu - \frac{1}{2}\sigma^2\right] t + \sigma W_{t_j}\right) \mid j = 0, 1, 2, \cdots, n\right\}$를 계산한다.

이 $\{x_{t_j} \mid j = 1, 2, \cdots, n\}$가 기하Brown운동의 표본경로이다.

예제 5.2.3 알고리즘 5.2.3를 사용해서 기하Brown운동의 표본경로를 발생시키기 위해서, 다음 MATLAB프로그램 GeometricBrownianMotion1D101.m을 실행해보자.

```
% ------------------------------------------------------------------------
%  Filename: GeometricBrownianMotion1D101.m
%  Generating 1-D Geometric Brownian Motion
%  Programmed by CBS
%-------------------------------------------------------------------------
```

```
clear all, close all, clc
T = 3;                              % Time in Seconds
Deltat = 1/120;
N = T/Deltat;                       % Number of Samples
tt = Deltat*(0:N-1)';
rand('twister',5489)
W = [0; cumsum(randn(N-1,1))]*sqrt(Deltat);
mu = 1                              % Drift parameter
sigma = 0.70                        % Diffusion parameter
GeoB = exp((mu - 1/2*sigma^2)*tt + sigma*W);
% Plotting
plot(tt,GeoB,'k-','linewidth',2);
set(gca,'fontsize',11,'fontweigh','bold')
hold on
plot([0 T],[1 1],'r-','linewidth',1.5);
xlabel('\bf t'), ylabel('\bf B_{t}','rotation',0)
hold off
saveas(gcf,'GeometricBrownianMotion1D101','epsc')
save('GeometricBrownianMotion1D101','W')
% End of program
% --------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 추세모수가 $\mu = 1$이고 확산모수가 $\sigma = 0.7$인 기하 Brown운동의 표본경로 $\{x(t_j)\}$를 출력한다. 이 표본경로가 그림 5.2.3에 그려져 있다. ■

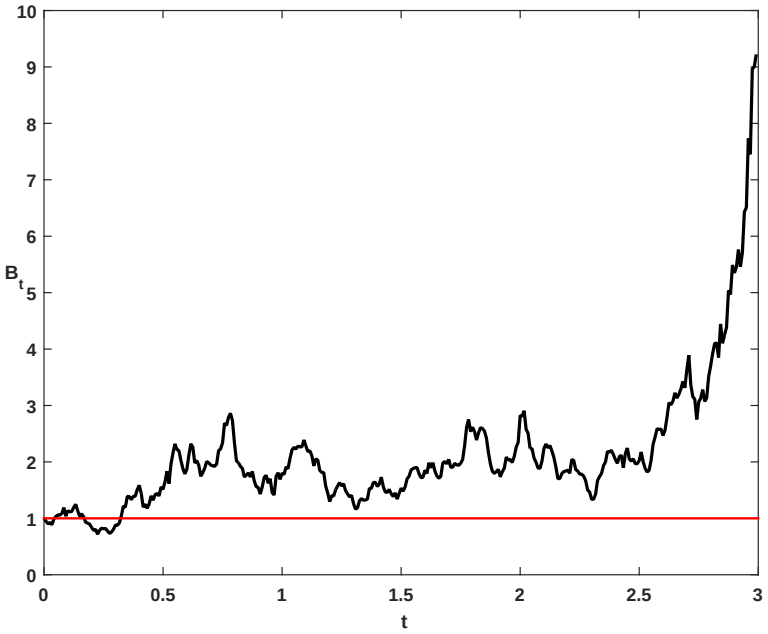


그림 5.2.3. 기하Brown운동의 표본경로

원자산이 기하Brown운동을 한다는 가정 하에 위험중립가치평가식을 사용해서 금융파생상품의 가치를 평가할 수 있다. 기하Brown운동의 표본경로들을 발생시켜 이들의 기대값을 구하는 예제를 살펴보자.

예제 5.2.4 기하Brown운동의 표본경로들을 발생시켜서 이들의 기대값을 구하기 위해서, 다음 MATLAB프로그램 GeometricBrownianMotion1D102.m을 실행해보자.

```
% ------------------------------------------------------------------------
%  Filename: Geometric BrownianMotion1D102.m
%  Generating 1-D Geometric Brownian Motions and their Mean
%  Programmed by CBS
%-------------------------------------------------------------------------
clear all, close all, clc
T = 3;                          % T years
dt = 1/252;                     % Time increment
N = T/dt;                       % Number of sampling times
tt = dt*(1:N)';                 % Sampling times
M = 100;                        % Number of sample paths
rng(5489,'twister')
dW = randn(N,M)*sqrt(dt);
W = cumsum(dW,1);
mu = 1                              % Drift parameter
sigma = 0.70                        % Diffusion parameter
X = exp((mu-1/2*sigma)*repmat(tt,[1,M]) + sigma*W);
Xmean = mean(X,2);
% Plotting
plot(tt,X(:,1:3),'linewidth',2);
set(gca,'fontsize',11,'fontweigh','bold')
hold on
plot(tt,Xmean,'k:','linewidth',3);
xlabel('\bf t'), ylabel('\bf W_{t}','rotation',0)
legend('Path 1','Path 2','Path 3','Mean Path','location','NW')
hold off
saveas(gcf,'GeometricBrownianMotion1D102','epsc')
save('GeometricBrownianMotion1D102','W')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 추세모수가 $\mu = 1$이고 확산모수가 $\sigma = 0.7$인 기하Brown운동의 표본경로들을 100개 생성하고, 또한 이 표본경로들의 표본평균을 계산한다. 이 표본경로들 중에서 첫 3개와 표본평균이 그림 5.2.4에 그려져 있다. ■

정의 5.2.4

표준Brown운동 $\{W_t\}$에 대해서 다음과 같이 정의되는 $\{b_t \mid 0 \le t \le T\}$를 시점 0에서 상태 α 부터 출발해서 시점 T에 상태 β에 도달하는 Brown다리과정(Brownian bridge)이라 부른다.

$$b_t^{\alpha\to\beta} \doteq \alpha\left[1-\frac{t}{T}\right] + \beta\frac{t}{T} + \left[W_t - \frac{t}{T}W_T\right], \quad (0 \le t \le T)$$

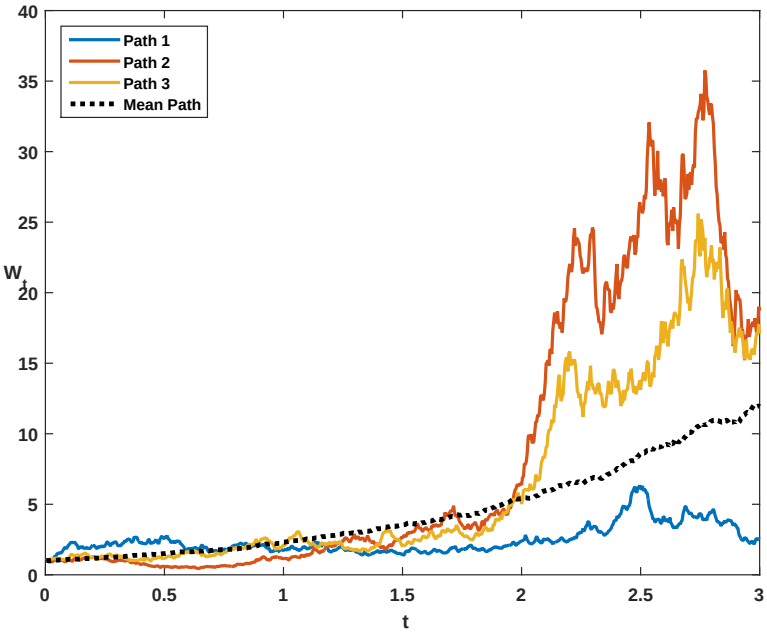


그림 5.2.4. 기하Brown운동의 표본경로들과 표본평균

정의 5.2.4에서 정의한 Brown다리과정의 평균과 분산은 각각 다음과 같다.

$$E\left(b_t^{\alpha\to\beta}\right) = \alpha\left[1-\frac{t}{T}\right]+\beta\frac{t}{T} \tag{5.2.4}$$

$$Cov\left(b_s^{\alpha\to\beta}, b_t^{\alpha\to\beta}\right) = s\left[1-\frac{t}{T}\right], \quad (0\le s\le t\le T) \tag{5.2.5}$$

알고리즘 5.2.4: Brown다리과정

(1단계) 표준Brown운동의 표본경로 $\left\{W_{t_j} \mid j=0,1,2,\cdots,n\right\}$를 구한다. 여기서 $W_{t_0}=W_0=0$이고, $t_n=T$이다.

(2단계) 수열 $\left\{b_{t_j}^{\alpha\to\beta} = \alpha\left[1-\frac{t_j}{T}\right]+\beta\frac{t_j}{T}+\left[W_{t_j}-\frac{t_j}{T}W_T\right] \;\middle|\; j=0,1,2,\cdots,n\right\}$를 계산한다.

이 $\left\{b_{t_j}^{\alpha\to\beta} \mid j=1,2,\cdots,n\right\}$가 Brown다리과정의 표본경로이다.

예제 5.2.5 알고리즘 5.2.4를 사용해서 Brown다리과정의 표본경로를 발생시키기 위해서, 다음 MATLAB프로그램 BrownianBridge1D101.m을 실행해보자.

```
% ------------------------------------------------------------------------
```

```
%  Filename: BrownianBridge1D101.m
%  Generating 1-D Brownian Bridge
%  Programmed by CBS
%--------------------------------------------------------------------------
clear all, close all, clc
T = 3;                              % Time in Seconds
Deltat = 1/120;
N = T/Deltat;                       % Number of Samples
tt = Deltat*(0:N)';
rand('twister',5489)
W = [0; cumsum(randn(N,1))]*sqrt(Deltat);
alpha = 1, beta = 2
BB = alpha*(1-tt/T) + beta*tt/T + (W - tt/T*W(end));
% Plotting
plot(tt,BB,'k-','linewidth',2);
set(gca,'fontsize',11,'fontweigh','bold')
hold on
plot([0 T],[alpha beta],'r-','linewidth',1.5);
plot(0,alpha,'bo',T,beta,'bo','linewidth',2);
xlabel('\bf t'), ylabel('\bf b_{t}','rotation',0)
hold off
saveas(gcf,'BrownianBridge1D101','epsc')
save('BrownianBridge1D101','BB')
% End of program
% --------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, Brown다리과정의 표본경로 $\left\{b_{t_j}^{1\to 2}\right\}$를 출력한다. 이 표본경로가 그림 5.2.5에 그려져 있다. ■

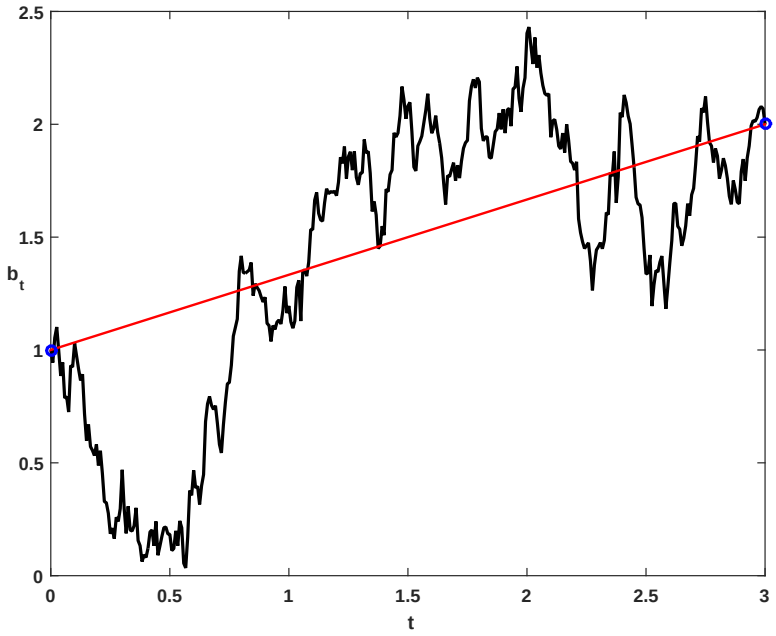


그림 5.2.5. Brown다리과정의 표본경로

알고리즘 5.2.1을 쉽게 다변량 Brownian운동으로 확장할 수 있다. 서로 독립인 표준Brown

운동들 $\{W_{1,t}\}, \{W_{2,t}\}, \cdots, \{W_{d,t}\}$에 대해서 확률벡터과정 $\{\boldsymbol{W}_t \doteq [W_{1,t}, W_{2,t}, \cdots, W_{d,t}]^t\}$를 d차원 표준Brown운동이라 한다. 만약 $d\boldsymbol{W}_t \doteq [dW_{1,t}, dW_{2,t}, \cdots, dW_{d,t}]^t$의 분산공분산행렬이 $\Sigma\, dt$이고 $\boldsymbol{W}_0 = \boldsymbol{0}$이면, $\{\Sigma^{-1/2}\boldsymbol{W}_t\}$가 d차원 표준Brown운동이라는 성질을 사용해서 다음과 같이 다변량 Brown운동에서 표본경로를 발생시킬 수 있다.

알고리즘 5.2.5: 다변량 Brown운동

(1단계) 양정치행렬 Σ에 대해서 식 $\Sigma = \Sigma^{1/2}\Sigma^{1/2}$인 양정치행렬 $\Sigma^{1/2}$를 구한다.

(2단계) 우선 d차원 표준Brown운동으로부터 표본경로의 차분 $\left\{\Delta\boldsymbol{W}_{t_j} \,\middle|\, j = 1, 2, \cdots, n\right\}$을 생성하자.

이 $\{\Delta\boldsymbol{W}_{t_j}\}$를 이용해서 생성한 확률과정 $\left\{\sum_{k=1}^{j} \Sigma^{1/2}\Delta\boldsymbol{W}_{t_k}\right\}$는 각 시점 t_j에서 분산공분산행렬이 $t_j\Sigma$인 d차원 Brown운동이다.

예제 5.2.6 알고리즘 5.2.5를 사용해서, 다음 식을 만족하는 의 표본경로 $\{\boldsymbol{W}_t\}$를 발생시키기로 하자.

$$\Delta\boldsymbol{W}_{t_j} = [t_{j+1} - t_j] \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}, \quad (j = 1, 2, \cdots) \tag{1}$$

이러한 표본경로를 발생시키기 위해서, 다음 MATLAB프로그램 BrownianMotion2D101.m을 실행해보자.

```
% ------------------------------------------------------------------------
%  Filename: BrownianMotion2D101.m
%  Generating 2-D Brownian Motion for T seconds
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all, clc
Sig2 = [ 1 0.5; 0.5 1]
SigHalf = sqrtm(Sig2)
T = 3;                              % Time in Seconds
Deltat = 1/120;
N = T/Deltat;                        % Number of Samples
tt = Deltat*(1:N)';
sqDeltat = sqrt(Deltat);
rand('twister',5489)
W = zeros(N,2);
for ii = 1:N-1
        W(ii+1,:) = W(ii,:) + sqDeltat.*randn(1,2)*SigHalf;
end
% Plotting
plot(W(:,1),W(:,2),'k-','linewidth',2);
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf W_{1,t}')
```

```
ylabel('\bf W_{2,t}','rotation',0)
grid on
axis equal
saveas(gcf,'BrownianMotion2D101','epsc')
save('BrownianMotion2D101','W')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB 프로그램을 실행하면, 식 (1)을 만족하는 2변량 Brown운동의 표본경로 $\{\boldsymbol{W}_{t_j}\}$를 출력한다. 이 표본경로가 그림 5.2.6에 그려져 있다. ■

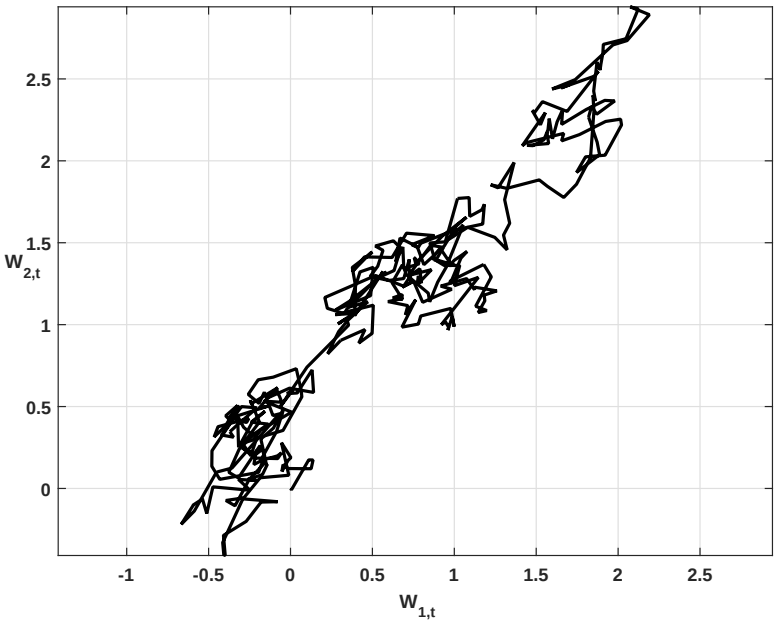


그림 5.2.6. 2변량 Brown운동의 표본경로

예제 5.2.7 알고리즘 5.2.5를 사용해서 다음 식을 만족하는 의 표본경로 $\{\boldsymbol{W}_t\}$를 발생시키기로 하자.

$$\Delta \boldsymbol{W}_{t_j} = [t_{j+1} - t_j] \begin{bmatrix} 1 & 0.25 & 0 \\ 0.25 & 1 & 0.25 \\ 0 & 0.25 & 0 \end{bmatrix}, \quad (j = 1, 2, \cdots) \tag{1}$$

이러한 표본경로를 발생시키기 위해서, 다음 MATLAB 프로그램 BrownianMotion3D101.m을 실행해보자.

```
% ------------------------------------------------------------------------
%  Filename: BrownianMotion3D101.m
%  Generating 3-D Brownian Motion for T seconds
%  Programmed by CBS
%------------------------------------------------------------------------
```

```
clear all, close all, clc
Sig3 = [ 1 0.25 0 ; 0.25 1 0.25 ; 0 0.25 1]
SigHalf = sqrtm(Sig3)
T = 3;                              % Time in Seconds
Deltat = 1/120;
N = T/Deltat;                       % Number of Samples
tt = Deltat*(1:N)';
sqDeltat = sqrt(Deltat);
rand('twister',5489)
W = zeros(N,3);
for ii = 1:N-1
        W(ii+1,:) = W(ii,:) + sqDeltat.*randn(1,3)*SigHalf;
end
% Plotting
plot3(W(:,1),W(:,2),W(:,3),'k-','linewidth',2);
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('\bf W_{1,t}'),ylabel('\bf W_{2,t}')
zlabel('\bf W_{3,t}','rotation',0)
grid on
saveas(gcf,'BrownianMotion3D101','epsc')
save('BrownianMotion3D101','W')
% End of program
% --------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 식 (1)을 만족하는 3변량 Brown운동의 표본경로 $\{\boldsymbol{W}_{t_j}\}$를 출력한다. 이 표본경로가 그림 5.2.7에 그려져 있다. ■

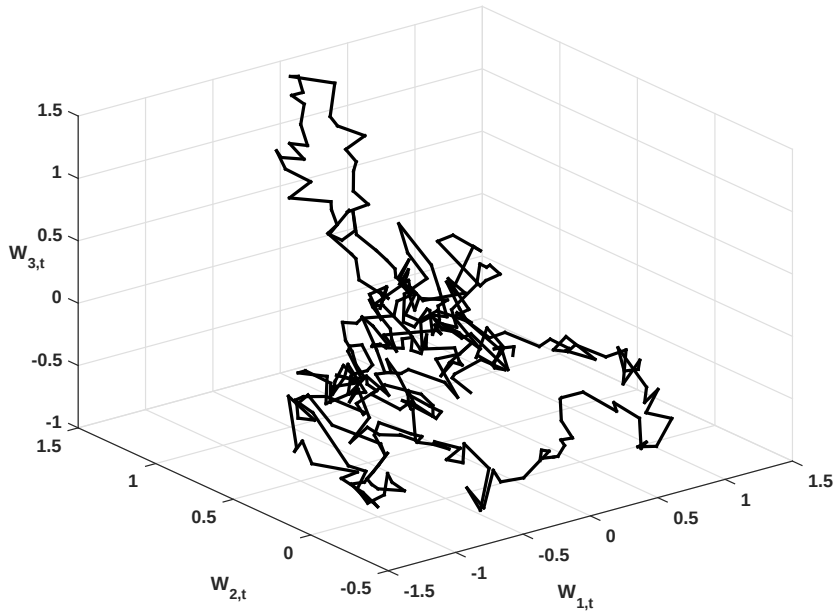


그림 5.2.7. 3변량 Brown운동의 표본경로

확률미분방정식이란 확률적분방정식을 간단하게 표기한 것이다. 따라서, 확률미분방정식을 풀기 위해서는 확률적분을 계산해야 한다. 그러나, 이 확률적분을 해석적으로 풀 수 있는 경우는 많지가 않다. 실제로 확률적분을 하기 위해서는 확률미분방정식을 만족하는

표본경로들을 생성한 다음, 대수법칙을 적용한다. 다음 예제는 확률적분을 계산하는 예이다.

예제 5.2.8 다음 Ito적분식이 성립한다.

$$\int_0^T W_t dW_t = \frac{1}{2}\left[W_T^2 - T\right] \tag{1}$$

이후 x_t를 $x(t)$로 표기하기도 할 것이다. Stratanovich적분식은 다음과 같이 정의된다.

$$\int_0^T W_t \circ dW_t \doteq \lim_{M\to\infty} \sum_{j=0}^{M-1} W\left(\frac{t_j + t_{j+1}}{2}\right)\left[W\left(t_{j+1}\right) - W\left(t_j\right)\right] \tag{2}$$

여기서 t_j는 다음과 같다.

$$t_j \doteq \frac{jT}{M}, \quad (j = 0, 1, \cdots, M) \tag{3}$$

다음 식이 성립한다.

$$\begin{aligned} 2W\left(\frac{t_j + t_{j+1}}{2}\right) = &\left[W\left(t_j\right) + W\left(t_{j+1}\right)\right] + \left[W\left(\frac{t_j + t_{j+1}}{2}\right) - W\left(t_{j+1}\right)\right] \\ &+ \left[W\left(\frac{t_j + t_{j+1}}{2}\right) - W\left(t_j\right)\right] \end{aligned} \tag{4}$$

또한, 다음 식이 성립한다.

$$W\left(t_{j+1}\right) - W\left(t_j\right) = \left[W\left(\frac{t_j + t_{j+1}}{2}\right) - W\left(t_j\right)\right] - \left[W\left(\frac{t_j + t_{j+1}}{2}\right) - W\left(t_{j+1}\right)\right] \tag{5}$$

식 (4)와 식 (5)를 식 (2)에 대입하면, 다음 식을 얻는다.

$$\begin{aligned} &\sum_{j=0}^{M-1} W\left(\frac{t_j + t_{j+1}}{2}\right)\left[W\left(t_{j+1}\right) - W\left(t_j\right)\right] \\ &= \frac{1}{2}\left[W^2(T) - W^2(0)\right] + \frac{\Delta t}{2}\frac{1}{2}\sum_{j=0}^{M-1}\left[z_{1,j}^2 - z_{2,j}^2\right] \end{aligned} \tag{6}$$

여기서 Δt, $z_{1,j}$ 와 $z_{2,j}$는 각각 다음과 같다.

$$\Delta t = \frac{T}{M} \tag{7}$$

$$z_{1,j} \doteq \sqrt{\frac{2}{\Delta t}} \left[W\left(\frac{t_j + t_{j+1}}{2}\right) - W\left(t_j\right) \right] \tag{8}$$

$$z_{2,j} \doteq \sqrt{\frac{2}{\Delta t}} \left[W\left(t_{j+1}\right) - W\left(\frac{t_j + t_{j+1}}{2}\right) \right] \tag{9}$$

확률벡터열 $\left\{[z_{1,j}, z_{2,j}]^t\right\}$은 서로 독립이고 평균벡터가 $\mathbf{0}$이고 분산공분산행열이 I인 2변량 정규확률분포를 따른다. 따라서 다음 식들이 성립한다.

$$E\left(\sqrt{\Delta t}\sum_{j=0}^{M-1}\left[z_{1,j}^2 - z_{2,j}^2\right]\right) = 0, \quad Var\left(\sqrt{\Delta t}\sum_{j=0}^{M-1}\left[z_{1,j}^2 - z_{2,j}^2\right]\right) = 6T \tag{10}$$

식 (6)과 식 (10)에서 알 수 있듯이, 다음 식이 성립한다.

$$\int_0^T W_t \circ dW_t = \frac{1}{2}W_T^2 \ \ in \ \ L^2 \tag{11}$$

Ito적분과 Stratanovich적분에 관해서는 최병선 [4]의 제3.4절을 참조하라.

Brown운동의 표본경로들을 생성해서 이 확률적분들을 계산하기 위해서, 다음 MATLAB 프로그램 StochasticIntegral1D101.m을 실행해보자.

```
% ------------------------------------------------------------------------
%  Filename: StochasticIntegral1D101.m
%  Ito integral and Stratonovich integral of WdW
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all, clc
T = 3;                          % T years
dt = 1/252;                     % Time increment
M = T/dt;                       % Number of sampling times
tt2 = dt/2*(1:2*M);             % Double sampling times
tt = tt2(2:2:2*M);
% Simulation
N = 100
rng(5489,'twister')
dW2 = randn(N,2*M)*sqrt(dt/2);
dW = dW2(:,1:2:2*M-1) + dW2(:,2:2:2*M);
W2 = cumsum(dW2);
Wodd = W2(:,1:2:2*M-1);
Weven = [ zeros(N,1), W2(:,2:2:2*(M-1)) ];
% Ito Integral
ItoI = sum(Weven.*dW,2);
ItoMean = mean(ItoI)
ItoStd = std(ItoI)
% Stratonovich Integral
StratonovichI = sum(Wodd.*dW,2);
```

```
StratonovichHist = hist(StratonovichI/T*2);
StratonovichMean = mean(StratonovichI)
StratonovichStd = std(StratonovichI)
% Histogram
subplot(211)
IxCenter = -2.5:0.25:0.5;
InBar = hist(ItoI,IxCenter);
InHeight = InBar/(IxCenter(2)-IxCenter(1))/N;
bar(IxCenter,InHeight,1,'w')
set(gca,'fontsize',11,'fontweigh','bold')
title('Ito Integral')
subplot(212)
SxCenter = (-2.5:0.25:0.5) +T/2;
SnBar = hist(StratonovichI,SxCenter);
SnHeight = SnBar/(SxCenter(2)-SxCenter(1))/N;
bar(SxCenter,SnHeight,1,'w')
set(gca,'fontsize',11,'fontweigh','bold')
title('Stratonovich Integral')
saveas(gcf,'StochasticIntegral1D101','epsc')
save('StochasticIntegral1D101','ItoI','StratonovichI')
% End of program
% --------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 시작 시점이 0이고 마지막 시점이 $T = 3$인 확률적분을 행한다. 표본경로들 100개 생성시켜 Ito적분 (1)을 추정한 표본평균은 다음과 같다.

$$\overline{\int_0^3 W_t dW_t} = -0.1135 \tag{12}$$

이에 해당하는 표본표준편차는 0.5830이다. 이 Ito적분의 기대값이 $E\left(\int_0^3 W_t dW_t\right) = 0$임을 고려하라. 또한, Stratonovich적분 (11)을 추정한 표본평균은 다음과 같다.

$$\overline{\int_0^3 W_t \circ dW_t} = 1.4575 \tag{13}$$

이에 해당하는 표본표준편차는 0.5780이다. 이 Stratonovich적분의 기대값이 $E\left(\int_0^3 W_t \circ dW_t\right) = \frac{3}{2}$임을 고려하라. 각 표본경로에 의해 계산된 확률적분의 히스토그램이 그림 5.2.8에 그려져 있다. ■

제5.3절 Markov체인

$\{x_t \mid t = 1, 2, \cdots, n\}$로부터 표본경로를 생성하는 과정은 간단하다. 우선, 주어진 확률분포에서 x_1을 생성한다. 다음으로 각 $t(= 1, 2, \cdots, n-1)$에 대해서 x_t가 주어진 조건 하에서 x_{t+1}

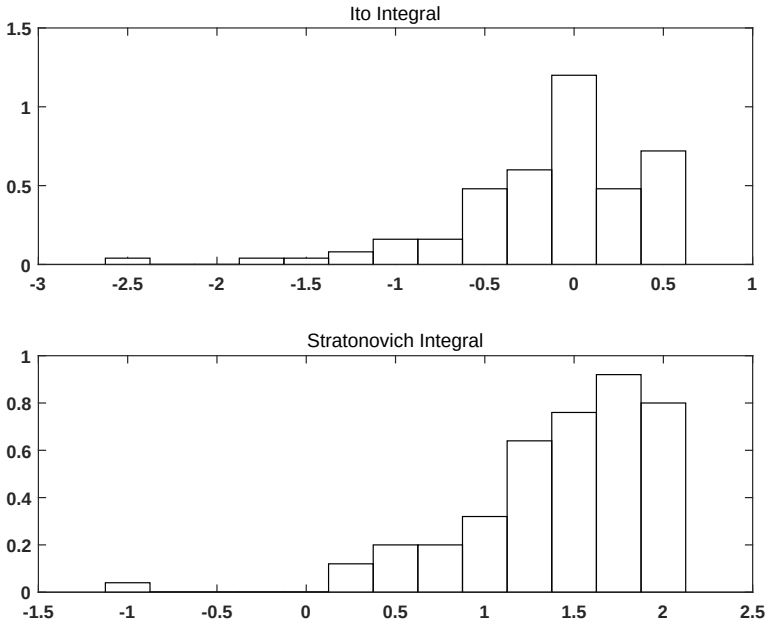


그림 5.2.8. Brown다리과정의 표본경로

의 조건부확률분포에서 x_{t+1}을 생성한다.

이산형 지지대가 유한집합 S인 Markov체인 $\{x_t \mid t = 1, 2, \cdots, n\}$의 $\mathbb{P} = [p_{ij}]$의 (i, j) 원소는 다음과 같다.

$$p_{ij} = Pr\left(x_{t+1} = j \mid x_t = i\right), \ \ (i, j \ \in S) \tag{5.3.1}$$

$x_{t+1} \mid x_t$의 는 추이행렬 $\mathbb{P}$의 제i번째 행이다. 따라서, 다음 알고리즘을 사용해서 이 Markov 연쇄로부터 표본경로를 발생시킬 수 있다.

알고리즘 5.3.1: Markov연쇄

(1단계) 초기확률분포로부터 x_1을 생성한다.

(2단계) 각 $t(= 1, 2, \cdots, n-1)$에 대해서, 추이행렬 $\mathbb{P}$의 x_t번째 행을 확률질량함수로 하는 확률분포에서 x_{t+1}을 생성한다.

이 $\{x_t\}$가 Markov체인의 표본경로이다.

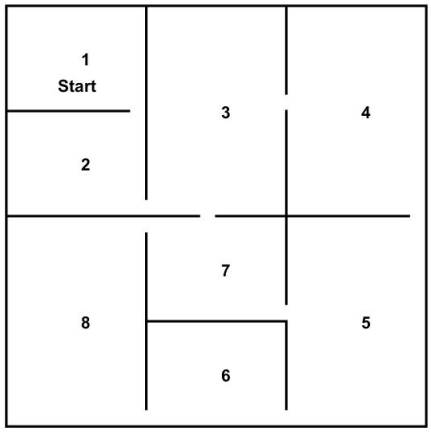
예제 5.3.1 로봇이 다니는 (maze)를 그리기 위해서 다음 MATLAB프로그램 MazeFigure101.m을 실행하라.

```
% --------------------------------------------------------------------------
% Filename: MazeFigure101.m
% Maze Figure for Markov Chain
% Programmed by CBS
% --------------------------------------------------------------------------
clear, clf
% Plotting
pa = 1/24;
hold on
plot([0 1 1 0 0],[0 0 1 1 0],'k','linewidth',1.5)
plot([0 1/3-pa],[3/4 3/4],'k','linewidth',1.5)
plot([0 1/2-pa],[2/4 2/4],'k','linewidth',1.5)
plot([1/3 1/3],[2/4+pa 1],'k','linewidth',1.5)
plot([1/3 1/3],[pa 2/4-pa],'k','linewidth',1.5)
plot([2/3 2/3],[pa 1/4],'k','linewidth',1.5)
plot([2/3 2/3],[1/4+pa 3/4],'k','linewidth',1.5)
plot([1/3 2/3],[1/4 1/4],'k','linewidth',1.5)
plot([1/2 1-pa],[2/4 2/4],'k','linewidth',1.5)
plot([2/3 2/3],[3/4+pa 1],'k','linewidth',1.5)
axis( [-0.2 1.2 -0.2 1.2  ])
axis square
axis off
set(gca,'fontsize',11,'fontweigh','bold')
text(1/9,13/16,'\bf Start')
text(1/6,7/8,'\bf 1'), text(1/6,5/8,'\bf 2')
text(1/6,2/8,'\bf 8'), text(3/6,6/8,'\bf 3')
text(3/6,3/8,'\bf 7'), text(3/6,1/8,'\bf 6')
text(5/6,6/8,'\bf 4'), text(5/6,2/8,'\bf 5')
hold off
saveas(gcf,'MazeFigure101','jpg')
% End of program
% --------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행시키면, 다음 그림이 그려진다.

시점 t에서 어떤 방에 있는 로봇은 다음 시점에 다른 방으로 이동한다고 하자. 단, 이동할 수 있는 각 방으로 가는 추이확률은 동일하다고 하자. 또한, 시점 1에서 로봇은 첫 번째 방에 있다고 하자. 예를 들어, 이 로봇이 시점 2에 두 번째 방으로 갈 확률은 1이다. 이 로봇이 시점 t에서 어느 방에 있느냐는 Markov체인를 이루고, 이 Markov체인의 추이행렬 $\mathbb{P}$는 다음과 같다.

$$\mathbb{P} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix} \tag{1}$$



시점 $t(=2,3,\cdots,60)$에서 이 로봇이 위치하는 방의 번호를 x_t 라 하고, 이 수열 $\{x_t\}$를 구하기 위해서 다음 MATLAB프로그램 Maze101.m을 실행해보자.

```
% ------------------------------------------------------------------------
% Filename: Maze101.m
% Maze for Markov Chain
% ------------------------------------------------------------------------
clear all, close all
rand('twister',5489);          % Use twister or state for seed.
Ns = 60;
P = [ 0   1   0   0  0   0   0  0
 1/2   0   1/2    0  0   0   0  0
0   1/3   0    1/3   0  0   1/3   0
0  0   1/2    0   1/2   0   0  0
0  0   0   1/3   0    1/3    1/3   0
0  0   0   0   1  0   0  0
0  0   1/3   0    1/3   0  0   1/3
0  0   0   0  0  1/2   1/2  0 ]
xx = zeros(Ns,1); tt = (1:Ns)';
xx(1) = 1;
for t=1:Ns-1
    dum = find(cumsum(P(xx(t),:)) > rand);
    xx(t+1) = min(dum);
end
% Plotting
plot(tt,xx,'k-',tt,xx,'rd','LineWidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[0 9])
saveas(gcf,'Maze101','epsc')
save('Maze101','xx')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 지지대가 $\{1, 2, 3, 4, 5, 6, 7, 8\}$ 인 Markov체인 $\{x_t\}$ 의 표본경로를 출력한다. 이 표본경로가 그림 5.3.1에 그려져 있다. ■

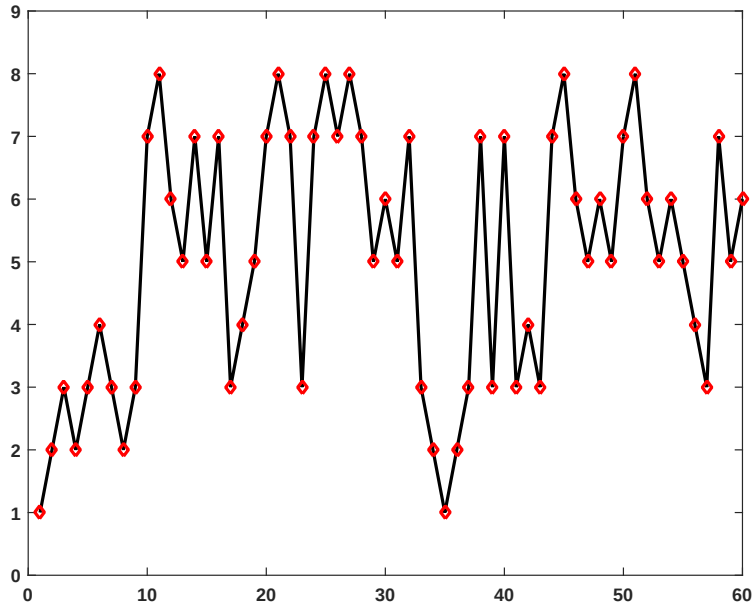
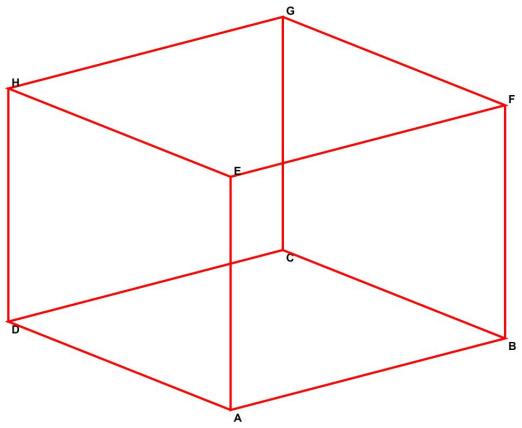


그림 5.3.1. 미로과정 $\{x_t\}$

예제 5.3.2 다음과 같은 정육면체를 살펴보자. 이 정육면체를 그리기 위해서는 다음 MATLAB프로그램 CubeFigure101.m을 실행하라.




```
% ------------------------------------------------------------------------
% Filename: CubeFigure101.m
% Cube Figure for Markov Chain
% Programmed by CBS
% ------------------------------------------------------------------------
clear, clf
```

```
% Plotting
x1 = [ 0 1 1 0 0 ];
y1 = [ 0 0 1 1 0 ];
z1 = [ 0 0 0 0 0 ];
z2 = [ 1 1 1 1 1 ];
hold on
plot3(x1,y1,z1,'r','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
plot3(x1,y1,z2,'r','linewidth',2)
view([-39 26])
plot3([0 0 1 1 0],[0 0 0 0 0],[1 0 0 1 1],'r','linewidth',2)
plot3([0 0 1 1 0],[1 1 1 1 1],[1 0 0 1 1],'r','linewidth',2)
d = 0.03;
text(0,0,0-d,'\bf A'), text(0,1,0-d,'\bf D')
text(1,0,0-d,'\bf B'), text(1,1,0-d,'\bf C')
text(0,0,1+d,'\bf E'), text(0,1,1+d,'\bf H')
text(1,0,1+d,'\bf F'), text(1,1,1+d,'\bf G')
axis off
hold off
saveas(gcf,'CubeFigure101','jpg')
% End of program
% ------------------------------------------------------------------------
```

어떤 스파이더로봇이 각 시점에서 이 정육면체의 꼭지점에 위치하고, 다음 시점에는 이웃하는 꼭지점으로만 이동한다. 예를 들어, 시점 t에 이 스파이더로봇이 꼭지점 A에 위치하고 있으면, 시점 $t+1$에는 꼭지점 B, 꼭지점 D 또는 꼭지점 E에 위치한다고 하자. 이때, 꼭지점 A에서 꼭지점 B, 꼭지점 D 또는 꼭지점 E으로 이동하는 확률은 동일하다. 시점 t에서 이 스파이더로봇의 위치를 열벡터 $\boldsymbol{x}_t$로 표기하자. 또한, 시점 1에서 위치는 $\boldsymbol{x}_1 = [0,0,0]^t$ 라고 하자. 또한, 다음과 같은 열벡터들을 정의하자.

$$\boldsymbol{e}_1 \doteq [1,0,0]^t, \quad \boldsymbol{e}_2 \doteq [0,1,0]^t, \quad \boldsymbol{e}_3 \doteq [0,0,1]^t \tag{1}$$

지지대가 $\{1,2,3\}$인 이산형 일양난수들을 $i_1, i_2, \cdots, i_n$이라 하자. 다음 식들이 성립함은 명백하다.

$$\boldsymbol{x}_{t+1} = [\boldsymbol{x}_t + \boldsymbol{e}_{i_t}] \pmod 2, \quad (t = 1,2,\cdots,n-1) \tag{2}$$

꼭지점을 나타내는 벡터 $\boldsymbol{x}_t = [x_{ti}, x_{t2}, x_{t3}]^t$에 대해서, 다음 함수를 생각해보자.

$$g(\boldsymbol{x}_t) = \frac{x_{1t}}{2} + \frac{x_{2t}}{2^2} + \frac{x_{3t}}{2^3} \tag{3}$$

이 함수 $g(\cdot)$가 1대1임은 명백하다. 따라서, 스칼라 $g(\boldsymbol{x}_t)$로부터 3변량 벡터 $\boldsymbol{x}_t = [x_{ti}, x_{t2}, x_{t3}]^t$를 일의적으로 구할 수 있다.

시점 $t(=2,3,\cdots,60)$에서 이 스파이더로봇이 위치하는 점열 $\{\boldsymbol{x}_t\}$을 구하기 위해서 다음

MATLAB프로그램 Cube101.m을 실행해보자.

```
% ------------------------------------------------------------------------
% Filename: Cube101.m
% Markov Chain for a Cube
% Programmed by CBS
% ------------------------------------------------------------------------
clear all, close all
rand('twister',5489);          % Use twister or state for seed.
Ns = 60;                       % Number of Samples
dim = 3;                       % Dimension of a Hypercube
xx = zeros(Ns,dim); tt = (1:Ns)';
for t=1:Ns-1
    dum = zeros(1,3);
    ind = ceil(3*rand);
    dum(ind) = 1;
    xx(t+1,:) = mod(xx(t,:)+dum,2);
end
bb = 2.^(-[1:dim]');
yy = xx*bb;
% Plotting
plot(tt,yy,'k-',tt,yy,'rd','LineWidth',2)
set(gca,'fontsize',11,'fontweigh','bold','ylim',[-0.1 1.1])
set(gca,'ytick',[0 1/8 1/4 3/8 1/2 5/8 3/4 7/8 ])
set(gca,'yticklabel',{'(0 0 0)' '(0 0 1)' '(0 1 0)' '(0 1 1)' ...
                      '(1 0 0)' '(1 0 1)' '(1 1 0)' '(1 1 1)'})
xlabel('\bf t')
saveas(gcf,'Cube101','epsc')
save('Cube101','xx','yy')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 지지대가 $\{A, B, C, D, E, F, G, H\}$ 인 Markov체인 $\{\boldsymbol{x}_t\}$의 표본경로를 출력한다. 이 표본경로가 그림 5.3.2에 그려져 있다. ■

제5.4절 확률미분방정식의 수치해법

5.4.1 확률미분방정식의 수치해

확률미분방정식의 명시적인 해를 구할 수 있는 경우는 아주 드물기 때문에, 현실에서는 시뮬레이션 기법들을 이용해서 근사해를 구하는 경우가 많다. 시뮬레이션 기법을 적용하는 주된 목적들 중의 하나는 해의 다양한 표본경로들을 시각적으로 관찰해서 해의 특징을 파악하는 데 있다. 이 때 얻어지는 표본경로들의 집합을 시나리오(scenario)라고 부르기도 한다. 이 시나리오를 바탕으로 확률과정의 예측값을 얻는 것이 가능하다. 현실세계에서 우리는 이산형 시간공간과 이산형 상태공간만을 다룰 수 있으므로, 결코 확률미분방정식의 근원이 되는

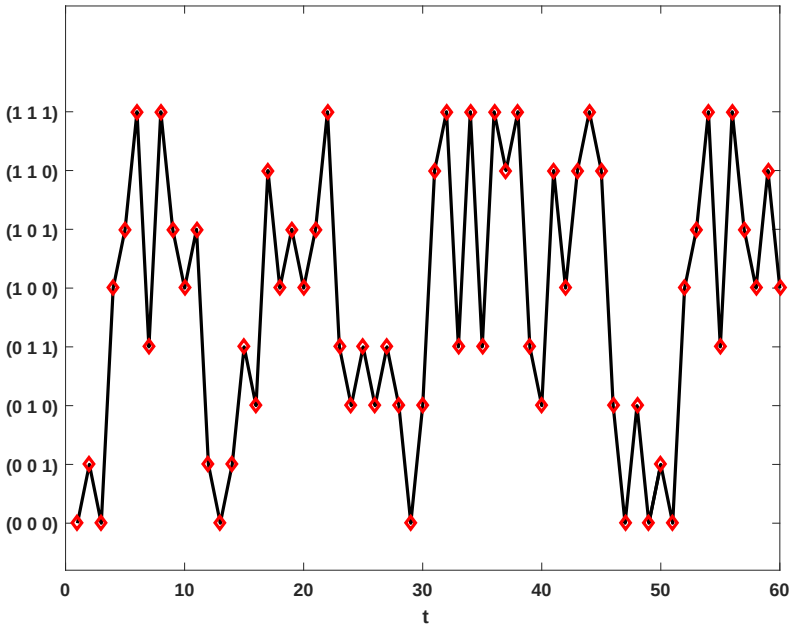


그림 5.3.2. 3변량 Markov체인 $\{\boldsymbol{x}_t\}$

Brown운동의 표본경로를 완전하게 파악할 수 없다. 즉, 시뮬레이트된 표본경로들이 항상 Brown운동의 일반적인 양상을 나타내고 있는 것은 아니므로, 시나리오를 채택할 때에는 충분한 주의를 기울여야 한다. 시뮬레이션 기법을 적용하는 다른 목적은 확률미분방정식의 해의 특성을 규정하는 기대값, 분산, 공분산, 고차적률 등을 근사적으로 도출하는 것이다. 일반적으로 이 통계량들에 대한 명시적인 식은 주어지지 않는다. 그러나, 명시적 해가 주어지는 경우에도 그 식 안에 수치적으로 근사계산을 하지 않으면 안되는 특수함수가 포함되는 경우가 많다. 충분한 시간을 가지고 몬테카를로법을 사용하면, 방대한 수의 표본경로들을 실현할 수 있기 때문에, 이러한 의미에서 몬테카를로법은 확률미분방정식을 수치적으로 다루는데 강력한 무기가 된다. 확률미분방정식의 수치적 해법에 관한 자세한 내용은 Kloeden & Platen [23], [24] 그리고 Asmussen & Glynn [9]을 참조하라.

다음 확률미분방정식을 살펴보자.

$$dx_t = \mu\left(x_t, t\right) dt + \sigma\left(x_t, t\right) dW_t, \quad (0 \le t \le T) \tag{5.4.1}$$

여기서 $\{W_t\}$는 Brown운동이다. 시간구간 $[0, T]$의 분할(partition) $\Pi \doteq \{0 = t_0 < t_1 < \cdots < t_M = T\}$가 다음 식을 만족한다고 하자.

$$\lim_{m \to \infty} \delta_M = 0 \tag{5.4.2}$$

여기서 $\Delta t_i \doteq t_{i+1} - t_i$이고, $\delta_M \doteq \max\limits_{0 \le i \le M-1} \Delta t_i$이다. 특별한 경우가 아니라면, 다음 식들을

만족하는 등간격인 분할을 사용하는 것이 편리하다.

$$t_{i+1} - t_i = \frac{T}{M}, \quad (i = 0, 1, \cdots, M-1) \tag{5.4.3}$$

확률미분방정식 (5.4.1)의 해를 $X_0^T \doteq \{x_t \mid 0 \le t \le T\}$ 라 하자. 각 $i(= 0, 1, \cdots, M)$에 대해서 시점 t_i에서 값 $x(t_i)$를 갖는 확률미분방정식 (5.4.1)의 해를 X라 하자. 이 경우에 각 시점 $t\,(\in (t_{i-1}, t_i))$에서는 확률미분방정식 (5.4.1)을 만족하는 x_t값을 구하는 것이 아니라, 이미 구해진 $\{x(t_i) \mid i = 0, 1, \cdots, M\}$에 보간법을 적용해서 x_t값을 구한다. 먼저 $\{x(t_i)\}$를 구하는 수치적 방법들을 살펴보기로 하자.

5.4.2 Euler-Maruyama근사법

확률미분방정식 (5.4.1)을 만족하는 표본경로 $\{x(t_i) \mid i = 0, 1, \cdots, M\}$을 구하는 가장 직관적인 방법은 확률미분방정식 (5.4.1)을 확률차분방정식으로 만드는 것이다. 이러한 과정을 Euler-Maruyama근사법 또는 Euler근사법이라 한다.

다음 확률변수들을 정의하자.

$$\Delta W(t_i) \doteq W(t_{i+1}) - W(t_i), \quad (i = 1, 2, \cdots, M) \tag{5.4.4}$$

확률미분방정식 (5.4.1)에 전진차분근사를 적용하면, 다음 점화식(recursive formula)(recursive formula)을 얻는다.

$$x(t_{i+1}) = x(t_i) + \mu(x(t_i), t_i)\,\Delta t_i + \sigma(x(t_i), t_i)\,\Delta W(t_i), \quad (i = 0, 1, \cdots, M-1) \tag{5.4.5}$$

단, 초기값 $x(0) = x_0$는 주어진 값이다.

예제 5.4.1 다음 기하Brown운동을 살펴보자.

$$dx_t = 2x_t dt + x_t dW_t, \quad (0 \le t \le T) \tag{1}$$

여기서 $T = 2$라고 하자. Euler-Maruyama법을 사용해서 확률미분방정식 (1)로부터 표본경로를 발생시키기 위해서, 다음 MATLAB프로그램 EulerMaruyama1D101.m을 실행해보자.

```
% ------------------------------------------------------------------------
%  Filename: EulerMaruyama1D101.m
```

```
%  Generating Geometric Brownian Motion using Euler-Maruyama method
%  Programmed by CBS
%-------------------------------------------------------------------------
clear all, close all, clc
mu = 1, sigma = 2           % Parameters
T = 2                       % Period in years
M = 250*T                   % Step numbers in [0 T]
dt = T/M                    % Time increment
Substeps = 40               % Substep numbers during one increment
ddt = dt/Substeps           % sub-increment
Mtotal = Substeps*M         % Total substeps
mu = 1, sigma = 2           % Parameters
T = 2                       % Period in years
M = 250*T                   % Step numbers in [0 T]
dt = T/M                    % Time increment
% Discretization of a true Brownian motion path
nSubsteps = 40              % Substep numbers during one increment
ddt = dt/nSubsteps          % sub-increment
Mtotal = nSubsteps*M        % Total nSubsteps
rng(5489,'twister')
Wtotal = cumsum(normrnd(0,sqrt(ddt),1,Mtotal));
W = [0,Wtotal];
dW = diff(W);
Anal0 = 1;
Anal = Anal0*exp((mu-0.5*sigma^2)*([0:ddt:T])+sigma*W);
whos
% Euler-Maruyama approximation with dt = 0.004
nSubsteps = 40
dEM1 = nSubsteps*ddt          % Increment of Euler-Maruyama method
nEM1 = Mtotal/nSubsteps;      % Number of Euler-Maruyama steps
xEM1 = zeros(1,nEM1);
xTemp = Anal0;
for ii = 1:nEM1
   Wincre = sum(dW(nSubsteps*(ii-1)+1:nSubsteps*ii));
   xTemp = xTemp + dEM1*mu*xTemp + sigma*xTemp*Wincre;
   xEM1(ii) = xTemp;
end
Error1 = Anal(end)-xEM1(end)
% Euler-Maruyama approximation with dt = 0.008
nSubsteps = 80
dEM2 = nSubsteps*ddt           % Increment of Euler-Maruyama method
nEM2 = Mtotal/nSubsteps;       % Number of Euler-Maruyama steps
xEM2 = zeros(1,nEM2);
xTemp = Anal0;
for ii = 1:nEM2
   Wincre = sum(dW(nSubsteps*(ii-1)+1:nSubsteps*ii));
   xTemp = xTemp + dEM2*mu*xTemp + sigma*xTemp*Wincre;
   xEM2(ii) = xTemp;
end
Error2 = Anal(end)-xEM2(end)
% Plotting
plot(0:ddt:T,Anal,'b-','linewidth',1);
set(gca,'fontsize',11,'fontweigh','bold')
hold on
plot(0:dEM1:T,[Anal0,xEM1],'black--','linewidth',2,'Markersize',5);
plot(0:dEM2:T,[Anal0,xEM2],'red:','linewidth',2);
xlabel('t'), ylabel('x_{t}','Rotation',0)
legend('discretized BM','Euler-Maruyama with dt = 0.04', ...
       'Euler-Maruyama with dt = 0.08','location','NE')
hold off
saveas(gcf,'EulerMaruyama1D101','epsc')
```

```
save('EulerMaruyama1D101','xEM1','xEM2')
% End of program
%--------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 식 (1)을 만족하는 기하Brown운동을 이산화한 (discretized) $\left\{ x\left(\frac{k}{250\cdot 40}\right) \;\middle|\; k = 0, 1, \cdots, 250\cdot 40\cdot T \right\}$를 생성한다. 또한, Euler-Maruyam법을 사용해서 표본경로 $\left\{ x\left(\frac{i}{250}\right) \;\middle|\; i = 0, 1, \cdots, 500 \right\}$ 와 표본경로 $\left\{ x\left(\frac{i}{125}\right) \;\middle|\; i = 0, 1, \cdots, 250 \right\}$를 생성한다. 마지막 시점 $T = 2$에서 이산화된 기하Brown운동의 실현값에서 시간간격이 0.004인 표본값을 뺀 값은 -0.0481이고, 시간간격이 0.008 표본값을 뺀 값은 -0.1512이다. 이렇게 생성된 이산화된 기하Brown운동의 실현과정과 표본경로들이 그림 5.4.1에 그려져 있다. 이 그림에서 이산화된 Brown운동은 청색 실선으로, 시간간격이 0.004인 표본경로는 흑색 긴점선으로, 그리고 시간간격이 0.008인 표본경로는 적색 점선으로 그려져 있다. 이 그림에서 알 수 있듯이, 관찰시점들의 간격이 짧을수록 표본경로는 이산화된 기하Brown운동에 가깝다. ■

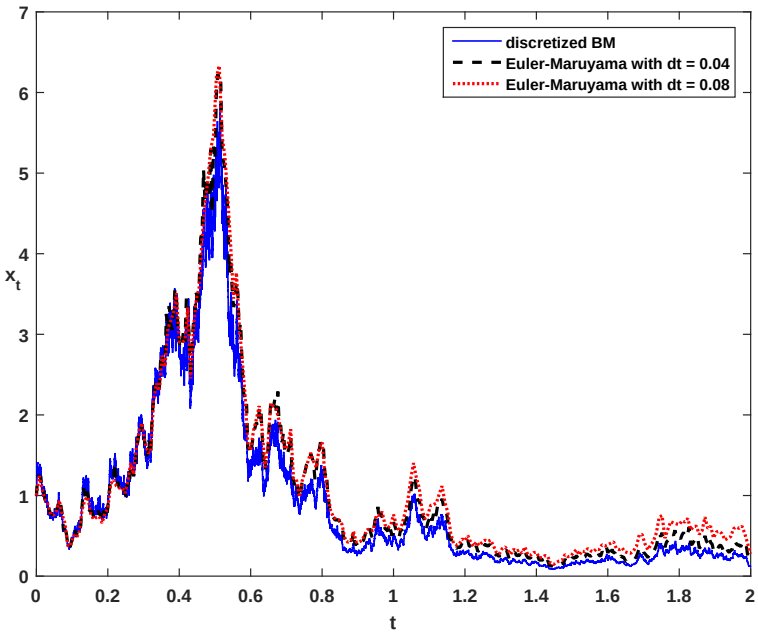


그림 5.4.1. Euler-Maruyama법에 의한 표본경로

점화식 (5.4.5)를 사용해서 생성한 $\{x(t_i) \mid i = 0, 1, \cdots, M\}$에 보간법을 적용해서 구한 확률미분방정식 (5.4.1)의 수치해 $\tilde{X} \doteq \{\tilde{x}_t \mid 0 \le t \le T\}$는 진짜해 $X \doteq \{x_t \mid 0 \le t \le T\}$와 가까워야 할 것이다. 그렇다면, 수치해 $\tilde{X}$가 진짜해 X에 가깝다는 것은 무엇을 의미할까? 또한, 이 가까움의 정도를 어떻게 정의할까? 진짜해 X와 수치해 $\tilde{X}$ 모두 사상(event) ω에

의존한다. 따라서, 진짜해와 수치해를 각각 $X(\omega)$ 와 $\tilde{X}(\omega)$로 표기하는 것이 더 명확할 것이다. 수치해 $\tilde{X}$가 진짜해 X의 근사성을 나타내는 대표적인 척도는 다음과 같다.

$$e_S(\delta_M) \doteq E(|\tilde{x}_T - x_T|) \tag{5.4.6}$$

여기서 첨자 S는 '강한(strong)' 이라는 의미이다. 또한, 다음과 같은 척도를 사용하기도 한다.

$$e_W(\delta_M) \doteq \sup_f E(|f(\tilde{x}_T) - f(x_T)|) \tag{5.4.7}$$

여기서 f는 다항식과 같은 매끄러운 함수이다. 식 (5.4.7)에서 첨자 W는 '약한(weak)' 이라는 의미이다. 물론, 다음과 같은 척도를 사용할 수도 있다.

$$e(\delta_M) \doteq E\left(\sup_{0\le t\le T} |\tilde{x}_t - x_t|\right) \tag{5.4.8}$$

그러나, 구간 $[0,T]$에서 모든 표본경로들을 다루어야 하는 $e(\delta_M)$을 분석하는 것은 매우 복잡한 문제이다.

정의 5.4.1

만약 확률미분방정식 (5.4.1)의 수치해 $\tilde{X} = \{\tilde{x}_t\}$가 다음 식을 만족하면, 이 $\tilde{X}$를 강한 수치해라 부른다.

$$\lim_{\delta_M \to 0} e_S(\delta_M) = 0$$

정의 5.4.2

만약 확률미분방정식 (5.4.1)의 수치해 $\tilde{X} = \{\tilde{x}_t\}$가 다음 식을 만족하면, 이 $\tilde{X}$를 약한 수치해라 부른다.

$$\lim_{\delta_M \to 0} e_W(\delta_M) = 0$$

다음과 같이 강수렴의 차수를 정의하자.

정의 5.4.3

각 δ_M에 대해서 다음 부등식을 만족하는 상수 $c_\gamma\,(>0)$가 존재하는 γ의 집합을 G_S라 하자.

$$e_S\left(\delta_M\right) \le c\delta_M^{\gamma}$$

수치해 $\tilde{X}$는 차수 $\max\{\gamma \in G_S\}$로 X에 강수렴(strong convergence)한다고 한다.

식 (5.4.2)가 성립하는 경우에, Euler-Maruyama근사는 차수 0.5로 강수렴함이 알려져 있다. 다음 예제는 이를 예증하기 위한 것이다.

예제 5.4.2 다음 기하Brown운동을 살펴보자.

$$dx_t = 3x_t dt + x_t dW_t, \quad (0 \le t \le T) \tag{1}$$

여기서 $T = 1$이라 하자. Euler-Maruyama근사가 차수 0.5로 강수렴하는 것을 확인하기 위해서, 다음 MATLAB프로그램 EulerMaruyamaStrongConvergence101.m을 실행해보자.

```
% ------------------------------------------------------------------------
%  Filename: EulerMaruyamaStrongConvergence101.m
%  Strong convergence of Euler-Maruyama method
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all, clc, format long
mu = 3, sigma = 1                   % Parameters
T = 1
Mtotal = 2^10                       % Number of total subintervals
ddt = T/Mtotal                      % Width of smallest subinterval
nPaths = 1000                       % Number of paths sampled
x0 = 1                              % Initial value
nSubintervals = 6                   % Number of subinterval widths
err = zeros(nPaths,nSubintervals);
rng(5489,'twister')                 % random seed
for jj = 1:nPaths
    dW = sqrt(ddt)*randn(1,Mtotal);% Brownian increments
    W = cumsum(dW);                 % discrete Brownian path
    discAnalT = x0*exp((mu-0.5*sigma^2)*T+sigma*W(end)); % x(T)
    for kk = 1:nSubintervals
        nSubsteps = 2^(kk-1);
        dt = nSubsteps*ddt;
        M = Mtotal/nSubsteps;
        xTemp = x0;
        for ii = 1:M
            Wincre = sum(dW(nSubsteps*(ii-1)+1:nSubsteps*ii));
            xTemp = xTemp + mu*xTemp*dt + sigma*xTemp*Wincre;
        end
        err(jj,kk) = abs(discAnalT-xTemp);       % Error at T
    end
```

```
end
% Regression Analysis; ln(mean(error)) = ln(C) + gamma*ln(dt)
dtVec = ddt*(2.^(0:(nSubintervals-1)))
log2dt = log2(dtVec)
tMax = max(ceil(log2dt))+0.3;
tMin = min(floor(log2dt))-0.3;
IndepVar = [ones(nSubintervals,1), log2dt']
DepVar = log2(mean(err,1))'
[bStrong,bint,rStrong,rint,stats] = regress(DepVar,IndepVar)
% Plotting
plot(log2dt,log2(mean(err)),'black*-','linewidth',2, ...
     'Markersize',10);
set(gca,'fontsize',11,'fontweigh','bold','xlim',[tMin,tMax])
hold on
plot(log2dt,bStrong(1)+bStrong(2)*log2dt,'red--','linewidth',2.5)
legend('Experimental Curve','Regression Line','location','NW')
xlabel('log_{2}(dt)'), ylabel('log_{2}(Error)')
hold off
saveas(gcf,'EulerMaruyamaStrongConvergence101','epsc')
save('EulerMaruyamaStrongConvergence101','xTemp','err')
% End of program
% --------------------------------------------------------------------------
```

정의 5.4.1에서 알 수 있듯이, 다음 식이 성립한다.

$$\log_2 e_S(\delta_M) \le \log_2 c + \gamma \log_2 \delta_M \tag{2}$$

이 MATLAB프로그램에서는 δ_M이 $2^{-10}, 2^{-9} \cdots, 2^{-6}, 2^{-5}$인 경우에 대해서 $e_S(\delta_M)$를 구하고, 이를 바탕으로 $\log_2 e_S(\delta_M)$이 $\log_2 \delta_M$의 선형함수이며 그 기울기가 0.5인지를 살펴본다.

이 MATLAB프로그램을 실행하면, 그림 5.4.2가 출력된다. 이 그림에서 흑색 별표로 표시된 점들이 $\{[\log_2 \delta_M, \log_2 e_S(\delta_M)]\}$이다. 이 그림에서 알 수 있듯이, $\log_2 e_S(\delta_M)$는 $\log_2 \delta_M$의 선형함수처럼 보인다. 이를 통계적으로 확인하기 위해서 회귀분석을 한 결과는 다음과 같다.

$$\log_2 e_S(\delta_M) = 4.5572 + 0.5998 \log_2 \delta_M + \epsilon_M \tag{3}$$

여기서 ϵ_M은 잔차이다. 또한, 이 선형회귀식의 R^2 값은 0.9952이고, F 값은 883.76이며, p 값은 0.000, 그리고 잔차들의 분산추정값은 0.0076이다. 따라서, $\log_2 e_S(\delta_M)$가 $\log_2 \delta_M$의 선형함수라고 결론지을 수 있다. 또한, γ의 추정값 0.5998은 0.5에 가까운 값이다.

본저자가 여러번 실험을 해본 결과, 조건 $\sigma^2 T > 1$이 성립하는 경우에, 위와 같은 결과가 나온다. 그렇지 않으면, $\log_2 e_S(\delta_M)$이 $\log_2 \delta_M$의 선형함수가 아니거나 선형함수라도 그 기울기가 0.5보다 큰 경우가 많았다. ■

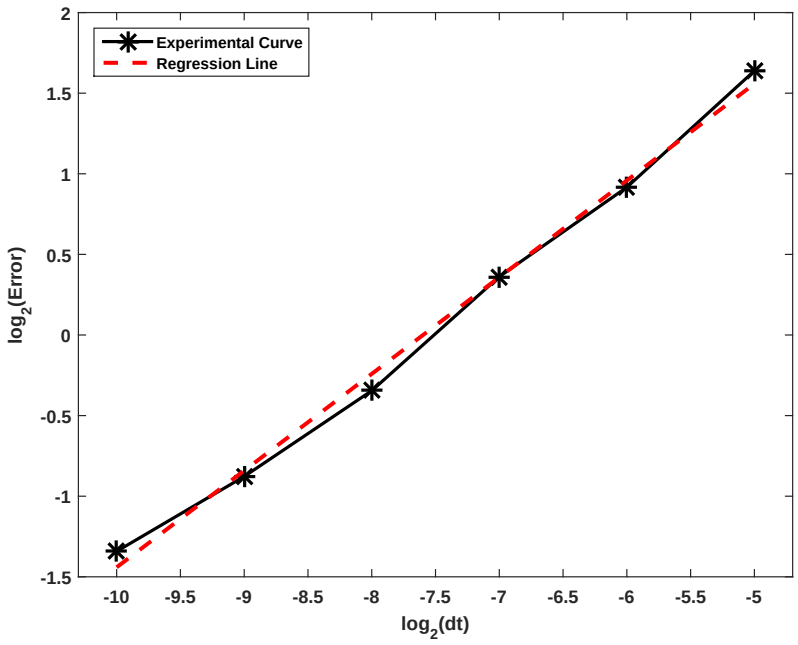


그림 5.4.2. Euler-Maruyama근사의 강수렴

정의 5.4.4

각 $\delta_M\,(\leq \delta_0)$에 대해서 다음 부등식을 만족하는 상수 $c_\gamma\,(>0)$가 존재하는 γ의 집합을 G_W라 하자.

$$e_W\,(\delta_M) \leq c\delta_M^{\gamma}$$

수치해 $\tilde{X}$는 차수 $\max\{\gamma \in G_W\}$로 X에 약수렴(weak convergence)(weak convergence) 한다고 한다.

식 (5.4.2)가 성립하는 경우에, Euler-Maruyama근사는 차수 1로 약수렴함이 알려져 있다. 다음 예제는 이를 예증하기 위한 것이다.

예제 5.4.3 예제 5.4.2에서와 마찬가지로 기하Brown운동을 살펴보자.

$$dx_t = 3x_t dt + x_t dW_t, \quad (0 \leq t \leq T) \tag{1}$$

여기서 $T = 1$이라고 하자. Euler-Maruyama근사가 차수 1로 약수렴하는 것을 확인하기 위해서, 식 (5.4.7)의 정의대신 다음 정의를 사용하기로 하자.

$$e_w\,(\delta_M) \doteq |\,E\,(\tilde{x}_T - x_T)\,| \tag{2}$$

확률미분방정식 (1)의 해는 다음과 같다.

$$x_T = x_0 \exp\left(\left[3 - \frac{1}{2}\cdot 1\right] T + \sigma W_T\right) \tag{3}$$

따라서, 다음 식이 성립한다.

$$E\left(x_T\right) = x_0 \exp\left(3T\right) \tag{4}$$

즉, 식 (2)를 다음과 같이 쓸 수 있다.

$$e_w\left(\delta_M\right) \doteq \mid E\left(\tilde{x}_T\right) - x_0 e^{3T} \mid \tag{5}$$

다음 MATLAB프로그램 EulerMaruyamaWeakConvergence101.m을 실행해보자.

```
% ------------------------------------------------------------------------
%  Filename: EulerMaruyamaWeakConvergence101.m
%  Weak convergence of Euler-Maruyama method
%  Programmed by CBS
%-------------------------------------------------------------------------
clear all, close all, clc
mu = 3, sigma = 1          % Parameters
T = 1
Mtotal = 2^10              % Number of total subintervals
ddt = T/Mtotal             % Width of the smallest subinterval
nPaths = 50000             % Number of paths sampled
x0 = 1                     % Initial value
nSubintervals = 6          % Number of subinterval widths
rng(5489,'twister')        % Random seed
for kk = 1:nSubintervals
    nSubsteps = 2^(kk-1);
    dt = nSubsteps*ddt;
    sqdt = sqrt(dt);
    M = Mtotal/nSubsteps;
    xTemp = x0*ones(nPaths,1);
    for ii = 1:M
        xTemp = xTemp + mu*xTemp*dt + sigma*sqdt*xTemp.* ...
                randn(nPaths,1);        ;
    end
    xEM(kk) = mean(xTemp);
end
err = abs(xEM - x0*exp(mu*T))'
% Regression Analysis; ln(mean(error)) = ln(C) + gamma*ln(dt)
dtVec = ddt*(2.^(0:(nSubintervals-1)))'
log2dt = log2(dtVec)
tMax = max(ceil(log2dt))+0.3;
tMin = min(floor(log2dt))-0.3;
IndepVar = [ones(nSubintervals,1), log2dt]
DepVar = log2(err)
whos
[bWeak,bint,rWeak,rint,stats] = regress(DepVar,IndepVar)
% Plotting
plot(log2dt,log2(err),'black*-','linewidth',2,'Markersize',10);
set(gca,'fontsize',11,'fontweigh','bold','xlim',[tMin,tMax])
```

```
hold on
plot(log2dt,bWeak(1)+bWeak(2)*log2dt,'red--','linewidth',2.5)
legend('Experimental Curve','Regression Line','location','NW')
xlabel('log_{2}(dt)'), ylabel('log_{2}(Error)')
hold off
saveas(gcf,'EulerMaruyamaWeakConvergence101','epsc')
save('EulerMaruyamaWeakConvergence101','xTemp','err')
% End of program
% --------------------------------------------------------------------------
```

식 (2)로부터 다음 식이 성립한다고 추측할 수 있다.

$$\log_2 e_w(\delta_M) \leq \log_2 c + \gamma \log_2 \delta_M \tag{6}$$

이 MATLAB프로그램에서는 δ_M이 $2^{-10}, 2^{-9} \cdots, 2^{-6}, 2^{-5}$인 경우에 대해서 $e_w(\delta_M)$를 구하고, 이를 바탕으로 $\log_2 e_w(\delta_M)$이 $\log_2 \delta_M$의 선형함수이며 그 기울기가 1인지를 살펴본다.

이 MATLAB프로그램을 실행하면, 그림 5.4.3이 출력된다. 그림 5.4.3에서 흑색 별표로 표시된 점들이 $\{[\log_2 \delta_M, \log_2 e_w(\delta_M)]\}$이다. 그림 5.4.3에서 알 수 있듯이, $\log_2 e_w(\delta_M)$는 $\log_2 \delta_M$의 선형함수처럼 보인다. 이를 통계적으로 확인하기 위해서 회귀분석을 한 결과는 다음과 같다.

$$\log_2 e_w(\delta_M) = 5.8693 + 0.9009 \log_2 \delta_M + \epsilon_M \tag{7}$$

여기서 ϵ_M은 잔차이다. 또한, 이 선형회귀식의 R^2 값은 0.9971이고, F 값은 1378.95이며, p 값은 0.000, 그리고 잔차들의 분산추정값은 0.0103이다. 따라서, $\log_2 e_w(\delta_M)$가 $\log_2 \delta_M$의 선형함수라고 결론지을 수 있다. 또한, γ의 추정값 0.9009는 1에 가까운 값이다. ■

5.4.3 확률적 Taylor근사

확률적 Taylor근사를 구하기 위해서, 우선 식 (5.4.1)을 다음과 같이 표기하자.

$$dx_t = \mu dt + \sigma dW_t \tag{5.4.9}$$

여기서 $\mu \doteq \mu(x_t, t)$이고 $\sigma \doteq \sigma(x_t, t)$이다. Euler-Maruyama근사식은 다음 식을 바탕으로 한 것이다.

$$\int_0^h \sigma(x_u, u)\, dW_u \simeq \sigma(x_0, 0)\, W_h \tag{5.4.10}$$

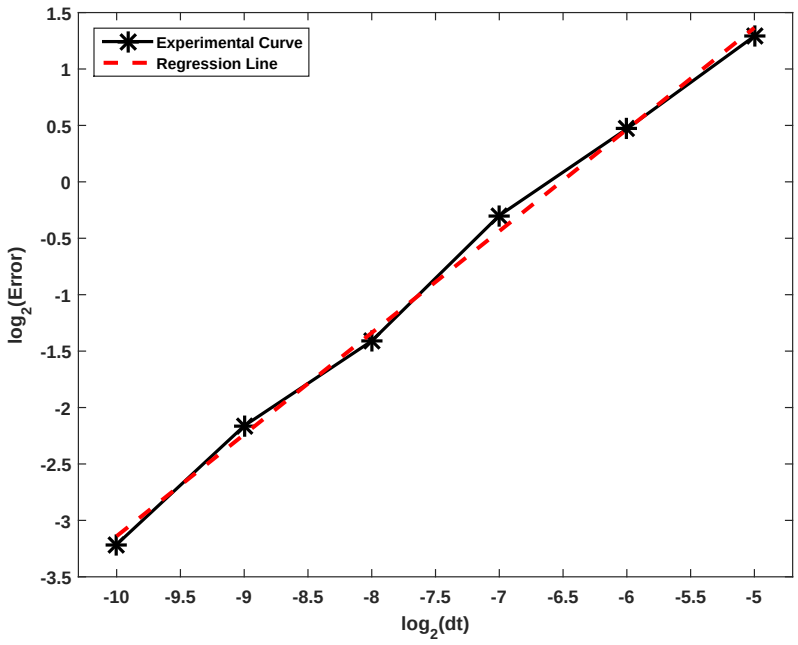


그림 5.4.3. Euler-Maruyama근사의 약수렴

좀 더 정교한 근사식을 구하기 위해서, 다음 식을 살펴보자.

$$\int_0^h \sigma\left(x_u, u\right) dW_u - \sigma\left(x_0, 0\right) W_h = \int_0^h \left[\sigma\left(x_u, u\right) - \sigma\left(x_0, 0\right)\right] dW_u \tag{5.4.11}$$

식 (5.4.11)의 우변에 Ito-Doeblin보조정리를 적용하면, 다음 식을 얻는다.

$$\begin{aligned} &\int_0^h \left[\sigma\left(x_u, u\right) - \sigma\left(x_0, 0\right)\right] dW_u \\ &= \int_0^h \left\{ \int_0^u \left[\sigma_u + \mu\sigma_x + \frac{1}{2}\sigma^2\sigma_{xx} \right] dv + \int_0^u \sigma\sigma_x dW_v \right\} dW_u \end{aligned} \tag{5.4.12}$$

Ito-Doeblin보조정리에 의해서 다음 식이 성립함을 알 수 있다.

$$d\left[\sigma\sigma_x\right] = \frac{\partial\left[\sigma\sigma_x\right]}{\partial t} dt + \frac{\partial\left[\sigma\sigma_x\right]}{\partial x} dx + \frac{1}{2} \frac{\partial^2\left[\sigma\sigma_x^2\right]}{\partial x} [dx]^2 \tag{5.4.13}$$

식 (5.4.13)에서 알 수 있듯이, 다음 식이 성립한다.

$$d\left[\sigma\sigma_x\right] = \frac{\partial\left[\sigma\sigma_x\right]}{\partial x} \sigma dW_z + o_p\left(dW_z\right) \tag{5.4.14}$$

즉, 다음 근사식이 성립한다.

$$\int_0^u \sigma\sigma_x dW_v \simeq [\sigma\sigma_x]_{t=0} \int_0^u dW_v + \int_0^u \int_0^v \sigma \frac{\partial [\sigma\sigma_x]}{\partial x} dW_z dW_v \tag{5.4.15}$$

식 (5.4.15)를 식 (5.4.12)에 대입하면, 다음 근사식을 얻는다.

$$\begin{aligned}
&\int_0^h [\sigma(x_u, u) - \sigma(x_0,\ 0)]\, dW_u \\
&\simeq \int_0^h \int_0^u \left[\sigma_u + \mu\sigma_x + \frac{1}{2}\sigma^2\sigma_{xx}\right] dv dW_u + [\sigma\sigma_x]_{t=0} \int_0^h \int_0^u dW_v dW_u \\
&\quad + \int_0^h \int_0^u \int_0^v \sigma \frac{\partial [\sigma\sigma_x]}{\partial x} dW_z dW_v dW_u
\end{aligned} \tag{5.4.16}$$

식 (5.4.16)을 정리하면, 다음과 같다.

$$\begin{aligned}
&\int_0^h [\sigma(x_u, u) - \sigma(x_0, 0)]\, dW_u \\
&\simeq \left[\sigma_t + \mu\sigma_x + \frac{1}{2}\sigma^2\sigma_{xx}\right]_{t=0} \int_0^h u dW_u + [\sigma\sigma_x]_{t=0} \int_0^h \int_0^u dW_v dW_u \\
&\quad + \int_0^h \int_0^u \int_0^v \sigma \frac{\partial [\sigma\sigma_x]}{\partial x} dW_z dW_v dW_u
\end{aligned} \tag{5.4.17}$$

즉, 다음 근사식이 성립한다.

$$\begin{aligned}
&\int_0^h [\sigma(x_u, u) - \sigma(x_0, 0)]\, dW_u \\
&\simeq \left[\sigma_t + \mu\sigma_x + \frac{1}{2}\sigma^2\sigma_{xx}\right]_{t=0} \cdot [hW_h - v_h] + [\sigma\sigma_x]_{t=0} \cdot \frac{1}{2}\left[W_h^2 - h\right] \\
&\quad + \sigma\left[\sigma\sigma_{xx} + \sigma_x^2\right]_{t=0} \cdot \int_0^h \left[\frac{1}{2}W_u^2 - \frac{u}{2}\right] dW_u
\end{aligned} \tag{5.4.18}$$

여기서 v_h는 다음과 같다.

$$v_h \doteq \int_0^h W_u du = hW_h - \int_0^h u dW_u \tag{5.4.19}$$

Ito-Doeblin보조정리에 의해서 다음 식이 성립함을 알 수 있다.

$$\int_0^h \left[\frac{1}{2}W_u^2 - \frac{u}{2}\right] dW_u = \frac{1}{2}\left[\frac{1}{3}W_h^3 - hW_h\right] \tag{5.4.20}$$

식 (5.4.20)을 식 (5.4.18)에 대입하면, 다음 근사식이 성립함을 알 수 있다.

$$\begin{aligned}
&\int_0^h \left[\sigma\left(x_u, u\right) - \sigma\left(x_0, 0\right)\right] dW_u \\
&\simeq \left[\sigma_t + \mu\sigma_x + \frac{1}{2}\sigma^2\sigma_{xx}\right]_{t=0} \cdot \left[hW_h - v_h\right] + \frac{1}{2}\left[\sigma\sigma_x\right]_{t=0}\left[W_h^2 - h\right] \\
&\quad + \frac{1}{2}\sigma\left[\sigma\sigma_{xx} + \sigma_x^2\right]_{t=0} \cdot \left[\frac{1}{3}W_h^3 - hW_h\right]
\end{aligned} \tag{5.4.21}$$

식 (5.4.21)을 유도하는 것과 비슷한 방법으로 다음 근사식을 유도할 수 있다.

$$\begin{aligned}
&\int_0^h \left[\mu\left(x_u, u\right) - \mu\left(x_0, 0\right)\right] dW_u \\
&\simeq \int_0^h \left\{\int_0^u \left[\mu_u + \mu\mu_x + \frac{1}{2}\sigma^2\mu_{xx}\right] dv + \int_0^u \sigma\mu_x dW_v\right\} du \\
&\simeq \frac{1}{2}\left[\mu_t + \mu\mu_x + \frac{1}{2}\sigma^2\mu_{xx}\right]_{t=0} h^2 + \sigma\mu_x \int_0^h W_u du
\end{aligned} \tag{5.4.22}$$

즉, 다음 근사식이 성립한다.

$$\int_0^h \mu\left(x_u, u\right) dW_u - \mu\left(x_0, 0\right) h \simeq \frac{1}{2}\left[\mu_t + \mu\mu_x + \frac{1}{2}\sigma^2\mu_{xx}\right]_{t=0} \cdot h^2 + \sigma\mu_x v_h \tag{5.4.23}$$

식 (5.4.21)과 식 (5.4.23)을 식 (5.4.9)에 대입하면, Ito-Taylor식을 얻는다.

$$\begin{aligned}
x_h =& x_0 + \mu h + \sigma W_h + \frac{1}{2}\left[\mu_t + \mu\mu_x + \frac{1}{2}\sigma^2\mu_{xx}\right]_{t=0} \cdot h^2 + \sigma\mu_x v_h \\
&+ \left[\sigma_t + \mu\sigma_x + \frac{1}{2}\sigma^2\sigma_{xx}\right]_{t=0} \cdot \left[hW_h - v_h\right] + \frac{1}{2}\left[\sigma\sigma_x\right]_{t=0}\left[W_h^2 - h\right] \\
&+ \frac{1}{2}\sigma\left[\sigma\sigma_{xx} + \sigma_x^2\right]_{t=0} \cdot \left[\frac{1}{3}W_h^3 - hW_h\right]
\end{aligned} \tag{5.4.24}$$

5.4.4 Milstein근사

Milstein근사식은 확률적 Taylor전개이론을 바탕으로 한 것이다. 편의상, 다음과 같은 시간 동질적 확률미분방정식을 살펴보자.

$$dx_t = \mu(x_t)dt + \sigma(x_t)dW_t, \quad (0 \le t \le T) \tag{5.4.25}$$

함수 $\sigma(x_t)$ 와 함수 $\mu(x_t)$ 에 Ito-Doeblin보조정리를 적용하면, 다음 식이 성립함을 알 수 있다.

$$x_t = x_s + \sigma(x_s)\int_s^t dW_u + \mu(x_s)\int_s^t du + R_{s,t} \tag{5.4.26}$$

여기서 잉여항 (remainder term) $R_{s,t}$ 는 다음과 같다.

$$\begin{aligned} R_{s,t} \doteq & \int_s^t dW_u \int_s^u \sigma(x_v)\sigma(x_v)dW_v \\ & + \int_s^t dW_u \int_s^u \left[\sigma(x_v)\mu(x_v) + \frac{1}{2}\sigma(x_v)\sigma^2(x_v)\right] dv \\ & + \int_s^t du \int_s^u \mu(x_v)\sigma(x_v)dW_v \\ & + \int_s^t du \int_s^u \left[\mu(x_v)\mu(x_v) + \frac{1}{2}\mu(x_v)\sigma^2(x_v)\right] dv \end{aligned} \tag{5.4.27}$$

식 (5.4.26) 에서 잉여항 $R_{s,t}$ 를 제외한 부분이 Euler-Maruyama근사식을 이룬다. 잉여항 $R_{s,t}$ 에 포함된 함수 $\sigma(x_v)\sigma(x_v)$ 에 Ito-Doeblin보조정리를 적용하면, 다음 식을 얻는다.

$$\sigma(x_v)\sigma(x_v) = \sigma(x_s)\sigma(x_s) + \int_s^v \frac{\partial\sigma(x_w))\sigma(x_w)}{\partial x_w}dx_w + \int_s^v \frac{\partial^2[\sigma(x_w))\sigma(x_w)]}{\partial x_w^2}\left[dx_w\right]^2 \tag{5.4.28}$$

식 (5.4.28) 을 식 (5.4.26) 에 대입하면, 다음 식을 얻는다.

$$x_t = x_s + \sigma(x_s)\int_s^t dW_u + \mu(x_s)\int_s^t du + \sigma(x_s)\sigma(x_s)\int_s^t dW_u \int_s^u dW_v + R^*_{s,t} \tag{5.4.29}$$

여기서 $R^*_{s,t}$ 는 새로운 잉여항이다. 식 (5.4.29) 에서 잉여항 $R^*_{s,t}$ 를 제외한 부분이 Milstein 근사식을 이룬다. 물론, 잉여항 $R^*_{s,t}$ 에 다시 Ito-Doeblin보조정리를 적용하면, 좀 더 정도 (precision) 가 높은 근사식을 얻을 수 있다.

식 (5.4.1) 에서 알 수 있듯이, 각 $i(=1,2,\cdots,M)$ 에 대해서 다음 식이 성립한다.

$$x\left(t_{i+1}\right) = x\left(t_i\right) + \int_{t_i}^{t_{i+1}} \mu\left(x_s, s\right) ds + \int_{t_i}^{t_{i+1}} \sigma\left(x_s, s\right) dW_s \tag{5.4.30}$$

Euler-Maruyama근사는 식 (5.4.30) 에 다음 근사식들을 적용한 것이다.

$$\int_{t_i}^{t_{i+1}} \mu(x_s, s)ds \simeq \mu\left(x^{(n)}\left(t_i\right), t_i\right)\Delta t_i \tag{5.4.31}$$

$$\int_{t_i}^{t_{i+1}} \sigma(x_s, s)dW_s \simeq \sigma\left(x^{(n)}\left(t_i\right), t_i\right)\Delta W\left(t_i\right) \tag{5.4.32}$$

즉, Euler-Maruyama근사는 다음과 같다.

$$x^{(n)}(t_{i+1}) = x^{(n)}(t_i) + \mu\left(x^{(n)}(t_i), t_i\right)\Delta t + \sigma\left(x^{(n)}(t_i), t_i\right)\Delta W\left(t_i\right) \tag{5.4.33}$$

다음 식들이 성립한다.

$$\begin{aligned}&\int_s^t dW_u \int_s^u dW_v = \int_s^t \left[W_u - W_s\right] dW_u \\ &= \frac{1}{2}\left[W_t^2 - W_s^2\right] - \frac{1}{2}[t-s] - W_s\left[W_t - W_s\right] = \frac{1}{2}\left[W_t - W_s\right]^2 - \frac{1}{2}[t-s]\end{aligned} \tag{5.4.34}$$

여기서 두 번째 등호는 Ito-Doeblin보조정리에 의해서 성립한다. 식 (5.4.34)를 식 (5.4.29)에 대입하면, 다음 식을 얻는다.

$$\begin{aligned}x_t = x_s &+ \sigma(x_s)\int_s^t dW_u + \mu(x_s)\int_s^t du \\ &+ \frac{1}{2}\sigma(x_s)\sigma(x_s)\left\{\frac{1}{2}\left[W_t - W_s\right]^2 - \frac{1}{2}[t-s]\right\} + R^*_{s,t}\end{aligned} \tag{5.4.35}$$

분할 Π에서 식 (5.4.35)를 적용하면, 다음 Milstein근사식을 얻는다.

$$\begin{aligned}x(t_{i+1}) = x(t_i) &+ \mu\left(x(t_i), t_i\right)\Delta t + \sigma\left(x(t_i), t_i\right)\Delta W\left(t_i\right) \\ &+ \frac{1}{2}\sigma\left(x(t_i), t_i\right)\frac{\partial\sigma\left(x(t_i), t_i\right)}{\partial x}\left\{\left[\Delta W\left(t_i\right)\right]^2 - \Delta t\right\}\end{aligned} \tag{5.4.36}$$

이 Milstein근사는 차수 1로 강수렴함이 알려져 있다.

다음 예제는 Milstein근사와 Euler-Maruyama근사를 비교하기 위한 것이다.

예제 5.4.4 다음 기하Brown운동을 살펴보자.

$$dx_t = x_t dt + 2x_t dW_t, \quad (0 \le t \le T) \tag{1}$$

여기서 $T = 2$라고 하자. 확률미분방정식 (1)로부터 표본경로를 발생시키기 위해서, 다음 MATLAB프로그램 Milstein1D101.m을 실행해보자.

```
% ------------------------------------------------------------------------
%  Filename: Milstein1D101.m
%  Generating 1-D Geometric Brownian Motion using Milstein Method
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all, clc
mu = 1, sigma = 2              % Parameters
```

```
T = 2                           % Period in years
dt = 1/250                      % Time increment
nSteps = T/dt                   % Time steps on [0 T]
subSteps = 40                   % Substep numbers in a time step
ddt = dt/subSteps               % sub-increment
Mtotal = subSteps*nSteps        % Total time substeps
% Generating Brownian paths
rng(5489,'twister')
wtot = cumsum(normrnd(0,sqrt(ddt),1,Mtotal));
w = [0,wtot(1,subSteps*(1:nSteps))];
dw = diff(w);
EM = ones(1,nSteps+1);          % Euler-Maruyama
Mils = ones(1,nSteps+1);        % Milstein
for ii = 1:nSteps
    EM(1,ii+1) = EM(1,ii) + mu*dt*EM(1,ii) + sigma*EM(1,ii)*dw(1,ii);
    Mils(1,ii+1) = Mils(1,ii) + Mils(1,ii)*(mu*dt + sigma*dw(1,ii)) ...
                   + 1/2*Mils(1,ii)*sigma^2*((dw(1,ii)^2 - dt));
end
% Plotting
xx = dt*[0,1:nSteps];
xAnal = ddt*(1:Mtotal);
Anal = exp(((mu-.5*sigma^2)*ddt*(1:Mtotal)) + sigma*wtot); % Analystic Sol
plot(xAnal,Anal,'g-',xx,EM,'r:',xx,Mils,'black--','linewidth',2);
set(gca,'fontsize',11,'fontweigh','bold')
legend('discretized Brown Motion','Euler-Maruyama','Milstein', ...
       'location','NE')
% Mean squared differences
AnalSub = [0 Anal(subSteps:subSteps:Mtotal)];
MSE  = [mean((AnalSub-EM).^2), mean((AnalSub-Mils).^2)]
saveas(gcf,'Milstein1D101','epsc')
save('Milstein1D101','EM','Mils')
% End of program
%-------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 확률미분방정식 (1)을 만족하는 기하Brown운동을 이산화한 (discretized) $\left\{x\left(\frac{k}{250\cdot 40}\right) \mid k=0,1,\cdots,250\cdot 40\cdot T\right\}$를 생성한다. 또한, Euler-Maruyam법에 의한 표본경로 $\left\{x_E\left(\frac{i}{250}\right) \mid i=0,1,\cdots,500\right\}$ 와 Milstein법에 의한 표본경로 $\left\{x_M\left(\frac{i}{250}\right) \mid i=0,1,\cdots,500\right\}$를 생성한다. 이렇게 생성된 이산화된 기하Brown운동의 실현과정과 표본경로들이 그림 5.4.4에 그려져 있다. 그림 5.4.4에서 이산화된 Brown운동은 녹색 실선으로, Euler-Maruyama법에 의한 표본경로는 적색 점선으로, 그리고 Milstein법에 의한 표본경로는 흑색 긴점선으로 그려져 있다. 그림 5.4.4에서 알 수 있듯이, Milstein법에 의한 표본경로가 Euler-Maruyama법에 의한 표본경로보다 더 이산화된 Brown운동에 더 가깝다. 이 예제에서 Euler-Maruyama법에 의한 표본경로와 이산화된 Brown운동의 평균제곱오차 (mean squared error)는 0.0526이고, Milstein법에 의한 표본경로와 이산화된 Brown운동의 평균제곱오차는 0.0023이다. ■

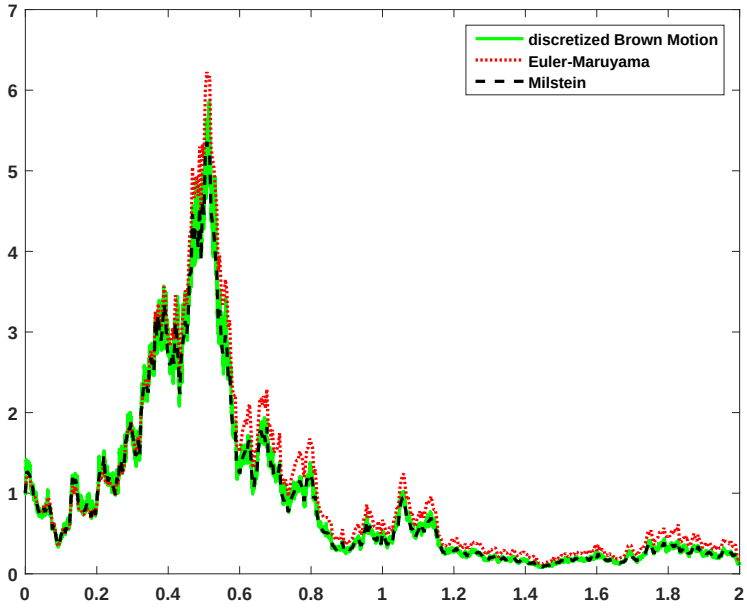


그림 5.4.4. Milstein법과 Euler-Maruyama법에 의한 표본경로들

제5.5절 확률미분방정식의 시뮬레이션과 MATLAB

5.5.1 확률미분방정식을 위한 MATLAB함수들

MATLAB의 Finance Toolbox에는 확률미분방정식(stochastic differential equation: SDE)의 몬테카를로실험을 수행해서 재무적 그리고 경제데이터를 분석하는 함수들을 포함하고 있다. 이 함수들은 유연한 구조를 가졌고, 따라서 이 함수들을 사용하면 효율적인 시뮬레이션방법으로 금융파생상품의 가치를 평가할 수 있다. 확률미분방정식의 모형화에 사용할 수 있는 MATLAB함수들은 SDE.m, SDEDDO.m, SDELD.m, SDEMRD.m, Heston.m, BM.m, CEV.m, HWV.m, CIR.m 그리고 GBM.m이 있다. 또한, 이들의 성분을 지정하는 Drift.m과 Diffusion.m이 있으며, 그 외에 이들과 연관된 simulate.m, simByEuler.m, interpolate.m, simBySolution이 있다. 이들 사이의 포함관계가 그림 5.5.1에 수록되어 있다. 이 절에서는 이 함수들을 사용하는 기본적인 방법을 다루기로 하자. 이 함수들에 대한 자세한 내용은 Finance Toolbox의 매뉴얼을 참조하라. 이 절의 내용은 다음 웹사이트의 내용을 바탕으로 한 것이다.

```
http://kr.mathworks.com/help/finance/stochastic-differential-
    equation-sde-models.html
```

[1] SDE함수

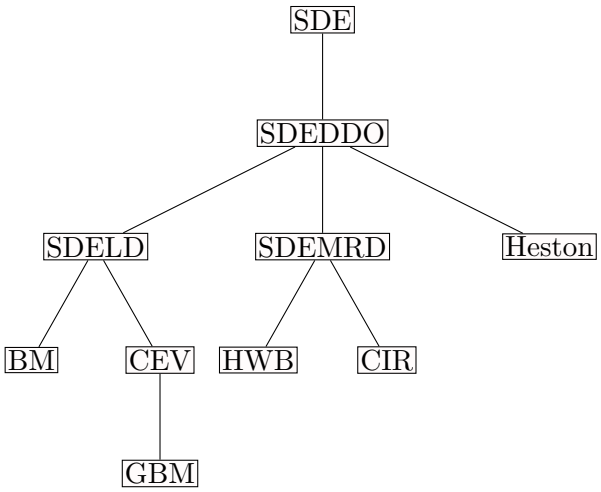


그림 5.5.1. 확률미분방정식을 위한 MATLAB함수들

가장 포괄적인 함수인 sde.m은 다음과 같은 SDE모형을 다루기 위한 것이다.

$$d\boldsymbol{x}_t = F(t, \boldsymbol{x}_t)\, dt + G(t, \boldsymbol{x}_t)\, d\boldsymbol{W}_t \tag{5.5.1}$$

여기서 $\boldsymbol{x}_t$는 차원 $NVARS \times 1$인 상태벡터이고, $\{\boldsymbol{W}_t\}$는 $NBROWNS$변량 Brown운동이고, 추세계수함수drift-rate function)추세계수함수인 $F(t, \boldsymbol{x}_t)$는 차원이 $NVARS \times 1$인 함수이고, 확산계수함수(diffusion-rate function)인 $G(t, \boldsymbol{x}_t)$는 $NVARS \times NBROWNS$행렬이다. 본서의 표기법대로라면 $F(t, \boldsymbol{x}_t)$ 와 $G(t, \boldsymbol{x}_t)$ 대신에 $F(\boldsymbol{x}_t, t)$ 와 $G(\boldsymbol{x}_t, t)$를 사용해야 하나, 이 절에서만은 Finance Toolbox의 매뉴얼의 표기법에 따라 $F(t, \boldsymbol{x}_t)$ 와 $G(t, \boldsymbol{x}_t)$를 사용하기로 하자. 이에 해당하는 MATLAB함수 sde.m의 사용법은 다음과 같다.

```
>> obj = sde(F,G)
```

예제 5.5.1 MATLAB함수 sde.m을 사용해서 SDE모형(SDE object)을 생성하는 예로서, 다음 MATLAB프로그램 UseSDE101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: UseSDE101.m
%  SDE object using MATLAB function sde.m
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all, clc
diary UseSDE101.txt
F1 = @(t,X) 0.1*t + 0.2*X
G1 = @(t,X) 0.3*10^(-7)*t + 0.4*X^0.3
obj1 = sde(F1,G1)
F2 = @(t,X) 0.1*sin(t) - 0.2*cos(X)
G2 = @(t,X) 0.3*cos(t) + 0.4*sin(X)
obj2 = sde(F2,G2)
```

```
diary off
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램은 다음과 같은 확률미분방정식을 모형화하기 위한 것이다.

$$dx_t = [0.2t + 0.1x_t]\,dt + [0.4t + 0.3x_t]\,dW_t \tag{1}$$

이 MATLAB프로그램을 실행하면, 다음과 같은 결과를 얻는다.

```
F =
    @(t,X)0.1*t+0.2*X

G =
    @(t,X)0.3*t+0.4*X

obj =
   Class SDE: Stochastic Differential Equation
   -------------------------------------------
     Dimensions: State = 1, Brownian = 1
   -------------------------------------------
      StartTime: 0
     StartState: 1
    Correlation: 1
          Drift: drift rate function F(t,X(t))
      Diffusion: diffusion rate function G(t,X(t))
     Simulation: simulation method/function simByEuler
```

MATLAB함수 sde.m에 의해 출력되는 모수들은 다음과 같다. 스칼라 StartTime은 초기시점(initial observation time)을 나타내고, 차원이 $NVARS \times 1$인 벡터 StartState는 초기상태벡터(initial state boldor)를 나타내고, 차원이 $NBROWNS \times NBROWNS$인 행렬 Correlation은 Brown운동들 사이의 상관계수행렬을 나타내고, $NVARS \times 1$인 벡터 Drift는 추세계수함수 $F(t, \boldsymbol{x}_t)$를 나타내고, 차원이 $NVARS \times NBROWNS$인 행렬 Diffusion은 확산계수함수 $G(t, \boldsymbol{x}_t)$를 나타내고, 모수 Simulation은 시뮬레이션방법이나 함수를 나타낸다. 이 중에서 Drift와 Diffusion만 필수적으로 사용해야 하는 입력모수들이다. MATLAB함수 sde.m에서는 모수 Correlation을 시간의 결정적 함수로 간주한다. 따라서, 이 모수를 사전에 Cholesky분해할 수 있고, 결과적으로 동적상관구조dynamic correlation structure)동적상관구조를 간단히 시뮬레이션할 수 있다. 확률적 상관계수구조를 갖는 확률미분방정식으로부터 시뮬레이션을 하기 위해서는 좀 더 일반적인 난수발생함수(more general random number generation function)를 사용해야 한다. ■

MATLAB함수 sde.m에서 추세계수함수의 모형을 지정하기 위해 MATLAB함수 drift.m 그리고 확산계수함수의 모형을 지정하기 위해 MATLAB함수 diffusion.m을 사용할 수 있다. MATLAB함수 drift.m에서 사용하는 추세계수모형은 다음과 같다.

$$F(t, \boldsymbol{x}_t) = A(t) + B(t)\boldsymbol{x}_t \tag{5.5.2}$$

여기서 $A(t)$는 차원이 $NVARS \times 1$인 함수이고, $B(t)$는 $NVARS \times NBROWNS$행렬이다. 이에 해당하는 MATLAB함수 drift.m의 사용법은 다음과 같다.

```
>> F = drift(A,B)
```

또한, MATLAB함수 diffusion.m에서 사용하는 확산계수모형은 다음과 같다.

$$G(t, \boldsymbol{x}_t) = D\left(t, \boldsymbol{x}_t^{\boldsymbol{\alpha}(t)}\right) V(t) \tag{5.5.3}$$

여기서 $\boldsymbol{\alpha}(t)$는 차원이 $NVARS \times 1$인 함수이고, $D\left(t, \boldsymbol{x}_t^{\boldsymbol{\alpha}(t)}\right)$는 차원이 $NVARS \times NVARS$이고 대각원소들이 $\boldsymbol{x}_t^{\boldsymbol{\alpha}(t)}$의 원소들인 대각행렬함수이고, $NVARS \times NBROWNS$행렬 $V(t)$는 변동성함수(volatility rate function)인 Sigma이다. 이에 해당하는 MATLAB함수 diffusion.m의 사용법은 다음과 같다.

```
>> G = diffusion(alpha,Sigma)
```

예제 5.5.2 MATLAB함수들 drift.m과 diffusion.m을 사용해서 추세계수모형과 확산계수모형을 지정하는 예로서, 다음 MATLAB프로그램 UseSDE102.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: UseSDE102.m
%  SDE object using MATLAB function sde.m, drift.m and diffusion.m
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all, clc
diary UseSDE102.txt
F = drift(0.1,0.2)
G = diffusion(0.3,0.4)
obj = sde(F,G)
diary off
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램은 다음과 같은 확률미분방정식을 모형화하기 위한 것이다.

$$dx_t = [0.1 + 0.2x_t]\,dt + 0.4x_t^{0.3}dW_t \tag{1}$$

이 MATLAB프로그램을 실행하면, 다음과 같은 결과를 얻는다.

```
F =
   Class DRIFT: Drift Rate Specification
   -------------------------------------
      Rate: drift rate function F(t,X(t))
         A: 0.1
         B: 0.2

G =
   Class DIFFUSION: Diffusion Rate Specification
   ---------------------------------------------
       Rate: diffusion rate function G(t,X(t))
      Alpha: 0.3
      Sigma: 0.4

obj =
   Class SDE: Stochastic Differential Equation
   -------------------------------------------
     Dimensions: State = 1, Brownian = 1
   -------------------------------------------
      StartTime: 0
     StartState: 1
    Correlation: 1
          Drift: drift rate function F(t,X(t))
      Diffusion: diffusion rate function G(t,X(t))
     Simulation: simulation method/function simByEuler
```

■

[2] SDEDDO함수

MATLAB함수 은 함수 sde.m으로부터 유도된 것으로, 다음과 같은 (SDE from drift and diffusion object)을 다루기 위한 것이다.

$$d\boldsymbol{x}_t = F(t, \boldsymbol{x}_t)\,dt + G(t, \boldsymbol{x}_t)\,d\boldsymbol{W}_t \tag{5.5.4}$$

여기서 $\boldsymbol{x}_t$는 차원 $NVARS \times 1$인 상태벡터이고, $\{\boldsymbol{W}_t\}$는 $NBROWNS$변량 Brown운동이고, 추세계수함수인 $F(t, \boldsymbol{x}_t)$는 차원이 $NVARS \times 1$인 함수이고, 확산계수함수인 $G(t, \boldsymbol{x}_t)$는 $NVARS \times NBROWNS$행렬이다. 이에 해당하는 MATLAB함수 sdeddo.m의 사용법은 다음과 같다.

```
>> obj = sdeddo(F,G)
```

함수 sdeddo.m이 함수 sde.m과 다른 점은 반드시 추세계수모형을 지정하기 위한 MATLAB 함수 drift.m과 확산계수모형을 지정하기 위한 MATLAB함수 diffusion.m을 함수 sdeddo.m 앞에서 사용해야한다는 것이다. 식 (5.5.2)와 식 (5.5.3)에서와 마찬가지로 식 (5.5.4)의 추세계수모형과 확산계수모형은 각각 다음과 같다.

$$F\left(t, \boldsymbol{x}_t\right) = A(t) + B(t)\boldsymbol{x}_t \tag{5.5.5}$$

$$G\left(t, \boldsymbol{x}_t\right) = D\left(t, \boldsymbol{x}_t^{\boldsymbol{\alpha(t)}}\right) V(t) \tag{5.5.6}$$

예제 5.5.3 MATLAB함수 sdeddo.m을 사용해서 SDE모형을 생성하는 예로서, 다음 MATLAB프로그램 UseSDEDD1O1.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: UseSDEDDO101.m
%  SDE object using Drift and Difusion objects by MATLAB function sdeddo.m
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all, clc
diary UseSDEDDO101.txt
F = drift(0.1,0.2)
G = diffusion(0.3,0.4)
obj = sdeddo(F,G)
diary off
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램은 다음과 같은 확률미분방정식을 모형화하기 위한 것이다.

$$dx_t = [0.1 + 0.2x_t]\, dt + 0.4x_t^{0.3} dW_t \tag{1}$$

이 MATLAB프로그램을 실행하면, 다음과 같은 결과를 얻는다.

```
F =
   Class DRIFT: Drift Rate Specification
   -------------------------------------
      Rate: drift rate function F(t,X(t))
         A: 0.1
         B: 0.2

G =
   Class DIFFUSION: Diffusion Rate Specification
   ---------------------------------------------
       Rate: diffusion rate function G(t,X(t))
```

```
        Alpha: 0.3
        Sigma: 0.4

obj =
   Class SDE: Stochastic Differential Equation
   -------------------------------------------
     Dimensions: State = 1, Brownian = 1
   -------------------------------------------
      StartTime: 0
     StartState: 1
    Correlation: 1
          Drift: drift rate function F(t,X(t))
      Diffusion: diffusion rate function G(t,X(t))
     Simulation: simulation method/function simByEuler
```

■

[3] SDELD함수

MATLAB함수 sdeld.m은 함수 sdeddo.m으로부터 유도된 것으로, 다음과 같은 SDELD 모형(SDE from linear drift object)을 다루기 위한 것이다.

$$d\boldsymbol{x}_t = [A(t) + B(t)\boldsymbol{x}_t]\,dt + D\left(t, \boldsymbol{x}_t^{\boldsymbol{\alpha}(t)}\right) V(t) d\boldsymbol{W}_t \tag{5.5.7}$$

여기서 $\boldsymbol{x}_t$는 차원 $NVARS \times 1$인 상태벡터이고, $\{\boldsymbol{W}_t\}$는 $NBROWNS$변량 Brown운동이고, $A(t)$는 차원이 $NVARS \times 1$인 함수이고, $B(t)$는 $NVARS \times NBROWNS$행렬이고, $\boldsymbol{\alpha}(t)$는 차원이 $NVARS \times 1$인 함수이고, $D\left(t, \boldsymbol{x}_t^{\boldsymbol{\alpha}(t)}\right)$는 차원이 $NVARS \times NVARS$이고 대각원소들이 $\boldsymbol{x}_t^{\boldsymbol{\alpha}(t)}$의 원소들인 대각행렬함수이고, $NVARS \times NBROWNS$행렬 $V(t)$는 변동성함수(volatility rate function)인 Sigma이다. 이에 해당하는 MATLAB함수 sdeld.m의 사용법은 다음과 같다.

```
>> obj = sdeld(A,B,alpha,Sigma)
```

함수 sdeld.m이 함수 sde.m과 다른 점은 추세계수모형을 지정하기 위한 MATLAB함수 drift.m이나 확산계수모형을 지정하기 위한 MATLAB함수 diffusion.m을 사용할 수 없다는 것이다.

예제 5.5.4 MATLAB함수 sdeld.m을 사용해서 SDE모형을 생성하는 예로서, 다음 MATLAB프로그램 UseSDELD101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: UseSDELD101.m
```

```
%  SDE from Linear Drift object by MATLAB function sdeld.m
%  Programmed by CBS
%--------------------------------------------------------------------------
clear all, close all, clc
diary UseSDELD101.txt
obj = sdeld(0.1,0.2,0.3,0.4)
diary off
% End of program
% --------------------------------------------------------------------------
```

이 MATLAB프로그램은 다음과 같은 확률미분방정식을 모형화하기 위한 것이다.

$$dx_t = [0.1 + 0.2x_t]\,dt + 0.4x_t^{0.3}dW_t \tag{1}$$

이 MATLAB프로그램을 실행하면, 다음과 같은 결과를 얻는다.

```
obj =
   Class SDELD: SDE with Linear Drift
   ----------------------------------------
     Dimensions: State = 1, Brownian = 1
   ----------------------------------------
      StartTime: 0
     StartState: 1
    Correlation: 1
          Drift: drift rate function F(t,X(t))
      Diffusion: diffusion rate function G(t,X(t))
     Simulation: simulation method/function simByEuler
              A: 0.1
              B: 0.2
          Alpha: 0.3
          Sigma: 0.4
```

■

[4] SDEMRD함수

MATLAB함수 sdemrd.m은 함수 sdeddo.m으로부터 유도된 것으로, 다음과 같은 SDEMRD 모형 (SDE from mean-reverting drift object)을 다루기 위한 것이다.

$$d\boldsymbol{x}_t = S(t)\,[L(t) - \boldsymbol{x}_t]\,dt + D\left(t, \boldsymbol{x}_t^{\boldsymbol{\alpha}(t)}\right) V(t) d\boldsymbol{W}_t \tag{5.5.8}$$

여기서 $\boldsymbol{x}_t$는 차원 $NVARS \times 1$인 상태벡터이고, $\{\boldsymbol{W}_t\}$는 $NBROWNS$ 변량 Brown운동이고, $S(t)$는 차원이 $NVARS \times NVARS$인 행렬함수로서 평균회귀속도 (mean reversion speed)를 나타내고, $L(t)$는 차원이 $NVARS \times 1$인 벡터함수로서 평균회귀수준 (mean reversion level)을 나타내고, $\boldsymbol{\alpha}(t)$는 차원이 $NVARS \times 1$인 벡터함수이고, $D\left(t, \boldsymbol{x}_t^{\boldsymbol{\alpha}(t)}\right)$

는 차원이 $NVARS \times NVARS$이고 대각원소들이 $\boldsymbol{x}_t^{\boldsymbol{\alpha}(t)}$의 원소들인 대각행렬함수이고, $NVARS \times NBROWNS$행렬 $V(t)$는 변동성함수(volatility rate function)인 Sigma이다. 이에 해당하는 MATLAB함수 sdemrd.m의 사용법은 다음과 같다.

```
>> obj = sdemrd(S,L,alpha,Sigma)
```

예제 5.5.5 MATLAB함수 sdemrd.m을 사용해서 SDE모형을 생성하는 예로서, 다음 MATLAB프로그램 UseSDEMRD101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: UseSDEMRD101.m
%  SDE from Mean-Reverting Drift object by MATLAB function sdemrd.m
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all, clc
diary UseSDEMRD101.txt
obj = sdemrd(0.1,0.2,0.3,0.4)
diary off
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램은 다음과 같은 확률미분방정식을 모형화하기 위한 것이다.

$$dx_t = 0.1\,[0.2 - x_t]\,dt + 0.4x_t^{0.3}dW_t \tag{1}$$

이 MATLAB프로그램을 실행하면, 다음과 같은 결과를 얻는다.

```
obj =

   Class SDEMRD: SDE with Mean-Reverting Drift
   -------------------------------------------
     Dimensions: State = 1, Brownian = 1
   -------------------------------------------
      StartTime: 0
     StartState: 1
    Correlation: 1
          Drift: drift rate function F(t,X(t))
      Diffusion: diffusion rate function G(t,X(t))
     Simulation: simulation method/function simByEuler
          Alpha: 0.3
          Sigma: 0.4
          Level: 0.2
          Speed: 0.1
```

■

[5] BM함수

MATLAB함수 bm.m은 함수 sdeld.m으로부터 유도된 것으로, 다음과 같은 BM모형(Brownian Motion object)을 다루기 위한 것이다.

$$d\boldsymbol{x}_t = \boldsymbol{\mu}(t)dt + V(t)d\boldsymbol{W}_t \tag{5.5.9}$$

여기서 $\boldsymbol{x}_t$는 차원 $NVARS \times 1$인 상태벡터이고, $\{\boldsymbol{W}_t\}$는 $NBROWNS$변량 Brown운동이고, $\boldsymbol{\mu}(t)$는 차원이 $NVARS \times 1$인 추세벡터함수이고, $NVARS \times NBROWNS$행렬 $V(t)$는 변동성함수인 Sigma이다. 이에 해당하는 MATLAB함수 bm101.m의 사용법은 다음과 같다.

```
>> obj = bm(mu,Sigma)
```

예제 5.5.6 MATLAB함수 bm.m을 사용해서 Brown운동모형을 생성하는 예로서, 다음 MATLAB프로그램 UseBM101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: UseBM101.m
%  Brownian Motion object by MATLAB function bm.m
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all, clc
diary UseBM101.txt
obj = bm(0.1,0.2)
diary off
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램은 다음과 같은 확률미분방정식을 모형화하기 위한 것이다.

$$dx_t = 0.1dt + 0.2dW_t \tag{1}$$

이 MATLAB프로그램을 실행하면, 다음과 같은 결과를 얻는다.

```
obj =
   Class BM: Brownian Motion
   ---------------------------------------
     Dimensions: State = 1, Brownian = 1
   ---------------------------------------
      StartTime: 0
     StartState: 0
    Correlation: 1
          Drift: drift rate function F(t,X(t))
      Diffusion: diffusion rate function G(t,X(t))
     Simulation: simulation method/function simByEuler
             Mu: 0.1
          Sigma: 0.2
```

■

[6] CEV함수

MATLAB함수 cev.m은 함수 sdeld.m으로부터 유도된 것으로, 다음과 같은 CEV모형(constant elasticity of variance: CEV object)을 다루기 위한 것이다.

$$d\boldsymbol{x}_t = \mu(t)\boldsymbol{x}_t dt + D\left(t, \boldsymbol{x}_t^{\boldsymbol{\alpha}(t)}\right) V(t) d\boldsymbol{W}_t \tag{5.5.10}$$

여기서 $\boldsymbol{x}_t$는 차원 $NVARS \times 1$인 상태벡터이고, $\{\boldsymbol{W}_t\}$는 $NBROWNS$변량 Brown운동이고, $\mu(t)$는 차원이 $NVARS \times NVARS$인 추세계소행렬함수을 나타내고, $\boldsymbol{\alpha}(t)$는 차원이 $NVARS \times 1$인 벡터함수이고, $D\left(t, \boldsymbol{x}_t^{\boldsymbol{\alpha}(t)}\right)$는 차원이 $NVARS \times NVARS$이고 대각원소들이 $\boldsymbol{x}_t^{\boldsymbol{\alpha}(t)}$의 원소들인 대각행렬함수이고, $NVARS \times NBROWNS$행렬 $V(t)$는 변동성함수인 Sigma이다. 이에 해당하는 MATLAB함수 sdemrd.m의 사용법은 다음과 같다.

```
>> obj = bm(mu,alpha,Sigma)
```

예제 5.5.7 MATLAB함수 cev.m을 사용해서 CEV모형을 생성하는 예로서, 다음 MATLAB프로그램 UseCEV101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: UseCEV101.m
%  Constant Elasticity of Variance object by MATLAB function cev.m
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all, clc
diary UseCEV101.txt
obj = cev(0.1,0.2,0.3)
diary off
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램은 다음과 같은 확률미분방정식을 모형화하기 위한 것이다.

$$dx_t = 0.1x_t dt + 0.3x_t^{0.2} dW_t \tag{1}$$

이 MATLAB프로그램을 실행하면, 다음과 같은 결과를 얻는다.

```
obj =
   Class CEV: Constant Elasticity of Variance
   ------------------------------------------
```

```
  Dimensions: State = 1, Brownian = 1
-----------------------------------------
   StartTime: 0
  StartState: 1
 Correlation: 1
       Drift: drift rate function F(t,X(t))
   Diffusion: diffusion rate function G(t,X(t))
  Simulation: simulation method/function simByEuler
      Return: 0.1
       Alpha: 0.2
       Sigma: 0.3
```

■

[7] GBM함수

MATLAB함수 gbm.m은 함수 cev.m으로부터 유도된 것으로, 다음과 같은 GBM모형(geometric Brownian motion: GBM object)을 다루기 위한 것이다.

$$d\boldsymbol{x}_t = \mu(t)\boldsymbol{x}_t dt + D\left(t, \boldsymbol{x}_t\right) V(t) d\boldsymbol{W}_t \tag{5.5.11}$$

여기서 $\boldsymbol{x}_t$는 차원 $NVARS \times 1$인 상태벡터이고, $\{\boldsymbol{W}_t\}$는 $NBROWNS$변량 Brown운동이고, $\mu\left(t\right)$는 차원이 $NVARS \times NVARS$인 추세계소행렬함수을 나타내고, $D\left(t, \boldsymbol{x}_t\right)$는 차원이 $NVARS \times NVARS$이고 대각원소들이 상태벡터 $\boldsymbol{x}_t$의 원소들인 대각행렬함수이고, $NVARS \times NBROWNS$행렬 $V\left(t\right)$는 변동성함수인 Sigma이다. 이에 해당하는 MATLAB함수 gbm.m의 사용법은 다음과 같다.

```
>> obj = gbm(mu,Sigma)
```

예제 5.5.8 MATLAB함수 gbm.m을 사용해서 GBM모형을 생성하는 예로서, 다음 MATLAB프로그램 UseGBM101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: UseGBM101.m
%  Geometric Brownian Motion object by MATLAB function gbm.m
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all, clc
diary UseGBM101.txt
obj = gbm(0.1,0.2)
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램은 다음과 같은 확률미분방정식을 모형화하기 위한 것이다.

$$dx_t = 0.1x_t dt + 0.2x_t dW_t \tag{1}$$

이 MATLAB프로그램을 실행하면, 다음과 같은 결과를 얻는다.

```
obj =
   Class GBM: Generalized Geometric Brownian Motion
   -------------------------------------------------
     Dimensions: State = 1, Brownian = 1
   -------------------------------------------------
      StartTime: 0
     StartState: 1
    Correlation: 1
          Drift: drift rate function F(t,X(t))
      Diffusion: diffusion rate function G(t,X(t))
     Simulation: simulation method/function simByEuler
         Return: 0.1
          Sigma: 0.2
```

■

[8] CIR함수

MATLAB함수 cir.m은 함수 sdemrd.m으로부터 유도된 것으로, 다음과 같은 CIR모형(Cox-Ingersoll-Ross object)을 다루기 위한 것이다.

$$d\boldsymbol{x}_t = S(t)\left[L(t) - \boldsymbol{x}_t\right]dt + D\left(t, \boldsymbol{x}_t^{1/2}\right)V(t)d\boldsymbol{W}_t \tag{5.5.12}$$

여기서 $\boldsymbol{x}_t$는 차원 $NVARS \times 1$인 상태벡터이고, $\{\boldsymbol{W}_t\}$는 $NBROWNS$변량 Brown운동이고, $S(t)$는 차원이 $NVARS \times NVARS$인 행렬함수로서 평균회귀속도를 나타내고, $L(t)$는 차원이 $NVARS \times 1$인 벡터함수로서 평균회귀수준을 나타내고, $D\left(t, \boldsymbol{x}_t^{1/2}\right)$는 차원이 $NVARS \times NVARS$이고 대각원소들이 $\boldsymbol{x}_t^{1/2}$의 원소들인 대각행렬함수이고, $NVARS \times NBROWNS$행렬 $V(t)$는 변동성함수인 Sigma이다. 이에 해당하는 MATLAB함수 cir.m의 사용법은 다음과 같다.

```
>> obj = cir(S,L,Sigma)
```

예제 5.5.9 MATLAB함수 cir.m을 사용해서 CIR모형을 생성하는 예로서, 다음 MATLAB프로그램 UseCIR101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: UseCIR101.m
%  Cox-Ingersoll-Ross object by MATLAB function cir.m
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all, clc
diary UseCIR101.txt
obj = cir(0.1,0.2,0.3)
diary off
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램은 다음과 같은 확률미분방정식을 모형화하기 위한 것이다.

$$dx_t = 0.1\,[0.2 - x_t]\,dt + 0.3x_t^{1/2}dW_t \tag{1}$$

이 MATLAB프로그램을 실행하면, 다음과 같은 결과를 얻는다.

```
obj =
   Class CIR: Cox-Ingersoll-Ross
   ----------------------------------------
     Dimensions: State = 1, Brownian = 1
   ----------------------------------------
      StartTime: 0
     StartState: 1
    Correlation: 1
          Drift: drift rate function F(t,X(t))
      Diffusion: diffusion rate function G(t,X(t))
     Simulation: simulation method/function simByEuler
          Sigma: 0.3
          Level: 0.2
          Speed: 0.1
```

■

[9] HWV함수

MATLAB함수 hwv.m은 함수 sdemrd.m으로부터 유도된 것으로, 다음과 같은 HWV모형(Hull-White/Vasicek object)을 다루기 위한 것이다.

$$d\boldsymbol{x}_t = S(t)\,[L(t) - \boldsymbol{x}_t]\,dt + V(t)d\boldsymbol{W}_t \tag{5.5.13}$$

여기서 $\boldsymbol{x}_t$는 차원 $NVARS \times 1$인 상태벡터이고, $\{\boldsymbol{W}_t\}$는 $NBROWNS$변량 Brown운동이고, $S(t)$는 차원이 $NVARS \times NVARS$인 행렬함수로서 평균회귀속도를 나타내고, $L(t)$는 차원이 $NVARS \times 1$인 벡터함수로서 평균회귀수준(mean reversion level)을 나타내고,

$NVARS \times NBROWNS$ 행렬 $V(t)$는 변동성의 변동성함수인 Sigma이다. 이에 해당하는 MATLAB함수 hwv.m의 사용법은 다음과 같다.

```
>> obj = hwv(S,L,Sigma)
```

예제 5.5.10 MATLAB함수 hwv.m을 사용해서 HWV모형을 생성하는 예로서, 다음 MATLAB프로그램 UseHWV101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: UseHWV101.m
%  Hull-White/Vasicek object by MATLAB function hwv.m
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all, clc
diary UseHWV101.txt
obj = hwv(0.1,0.2,0.3)
diary off
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램은 다음과 같은 확률미분방정식을 모형화하기 위한 것이다.

$$dx_t = 0.1\,[0.2 - x_t]\,dt + 0.3dW_t \tag{1}$$

이 MATLAB프로그램을 실행하면, 다음과 같은 결과를 얻는다.

```
obj =
   Class HWV: Hull-White/Vasicek
   ----------------------------------------
     Dimensions: State = 1, Brownian = 1
   ----------------------------------------
      StartTime: 0
     StartState: 1
    Correlation: 1
          Drift: drift rate function F(t,X(t))
      Diffusion: diffusion rate function G(t,X(t))
     Simulation: simulation method/function simByEuler
          Sigma: 0.3
          Level: 0.2
          Speed: 0.1
```

■

[10] Heston함수

MATLAB함수 heston.m은 함수 sdeddo.m으로부터 유도된 것으로, 다음과 같은 확률변동성모형인 Heston모형을 다루기 위한 것이다.

$$dx_{1,t} = B(t)x_{1,t}dt + \sqrt{x_{2,t}}x_{1,t}dW_{1,t} \tag{5.5.14}$$

$$dx_{2,t} = S(t)\left[L(t) - x_{2,t}\right]dt + V(t)\sqrt{x_{2,t}}dW_{2,t} \tag{5.5.15}$$

여기서 $x_{1,t}$ 와 $x_{2,t}$는 상태변수들이고, $\{W_{1,t}\}$ 와 $\{W_{2,t}\}$는 서로 독립인 표준Brown운동들이고, $S(t)$는 평균회귀속도를 나타내는 확률변수이고, $V(t)$는 변동성의 변동성함수인 Sigma이다. 이에 해당하는 MATLAB함수 heston.m의 사용법은 다음과 같다.

```
>> obj = heston(B,S,L,V)
```

예제 5.5.11 MATLAB함수 heston.m을 사용해서 Heston모형을 생성하는 예로서, 다음 MATLAB프로그램 UseHeston101.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: UseHeston101.m
%  Heston object by MATLAB function heston.m
%  Programmed by CBS
%------------------------------------------------------------------------
clear all, close all, clc
diary UseHeston101.txt
obj = heston(0.1,0.2,0.3,0.4)
diary off
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램은 다음과 같은 확률미분방정식을 모형화하기 위한 것이다.

$$dx_{1,t} = 0.1x_{1,t}dt + \sqrt{x_{2,t}}x_{1,t}dW_{1,t} \tag{1}$$

$$dx_{2,t} = 0.2\left[0.3 - x_{2,t}\right]dt + 0.4\sqrt{x_{2,t}}dW_{2,t} \tag{2}$$

이 MATLAB프로그램을 실행하면, 다음과 같은 결과를 얻는다.

```
obj =
   Class HESTON: Heston Bivariate Stochastic Volatility
   -----------------------------------------------------
     Dimensions: State = 2, Brownian = 2
   -----------------------------------------------------
      StartTime: 0
     StartState: 1 (2x1 double array)
    Correlation: 2x2 diagonal double array
          Drift: drift rate function F(t,X(t))
```

```
 Diffusion: diffusion rate function G(t,X(t))
Simulation: simulation method/function simByEuler
    Return: 0.1
     Speed: 0.2
     Level: 0.3
Volatility: 0.4
```

■

5.5.2 주가모형의 시뮬레이션

다음과 같은 분리 다변량 기하Brown운동과정 (separable multivariate geometric Brownian motion process)을 다변량 시장모형 (multidimensional market model)이라 부른다.

$$d\boldsymbol{x}_t = \mu \boldsymbol{x}_t dt + D(\boldsymbol{x}_t)\,\sigma d\boldsymbol{W}_t \tag{5.5.16}$$

여기서 μ는 기대수익률을 나타내는 대각행렬이고, $D(\boldsymbol{x}_t)$는 제i번째 대각원소가 $\boldsymbol{x}_t$의 제i번째 원소 $x_{i,t}$인 대각행렬이고, σ는 제i번째 주가수익률의 표준편차인 대각행렬이다. 이 다변량 시장모형에서 표본경로를 발생시키기 위해서, MATLAB함수들 sde, sdeddo, sdeddo, cev 그리고 gbm을 사용할 수 있다.

이 소절의 예제들은 데이터세트 Data_Global_Idx102를 바탕으로 주가를 시뮬레이션하는 것이다. 이 데이터세트는 2001년 2월 7일부터 2006년 4월 24일까지 캐나다 (TSX), 프랑스 (CAC), 독일 (DAX), 일본 (NIK), 영국 (FTSE), 미국 (SP)의 대형주의 지수들 (large-cap indexes)과 3개월물 Euribor(EB3M)의 일별데이터를 수록한 것이다.

예제 5.5.12 데이터세트 Data_Global_Idx102의 대형주지수들의 로그수익률을 계산하고, 요약통계량 (descriptive statistics)를 구하고, 이들을 저장한 새로운 MATLAB데이터세트를 만들고, 대형주지수들의 시계열산점도를 그리기 위해서, 다음 MATLAB프로그램 Data_Global_Idx102.m을 실행해보자.

```
% ------------------------------------------------------------------------
%  Filename: Data_Global_Idx102.m
%  Make Global_Idx102_Data.mat
%  Based on Chapter 8 of Econmtrics Toolbox Manual
%------------------------------------------------------------------------
clear all, close all, clc
load Data_GlobalIdx2
% Dataset(1:10,:)
prices = [ Dataset.TSX Dataset.CAC Dataset.DAX Dataset.NIK ...
           Dataset.FTSE Dataset.SP ];
```

```
euribor3M = Dataset.EB3M;
returns = price2ret(prices);
nalja = datestr(dates);
% Descriptive statistics for input to simulation methods
nVar = size(returns,2)
averReturn = mean(returns)
sigma = std(returns)
corrReturn = corr(returns)
% Save the dataset
% whos
save Global_Idx102_Data
% Plotting
plot(dates,prices,'linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[730889 732791])
DateTick = dates(1:252:end);
DateTickLabel = datestr(DateTick,2);
set(gca,'xtick',DateTick)
set(gca,'xticklabel',DateTickLabel)
legend('Canada','France','Germany','Japan','UK','US', ...
       'location','NW')
saveas(gcf,'GlobalIdx102.jpg')
figure
plot(dates(1:end-1),returns,'linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[730889 732791])
set(gca,'xtick',DateTick)
set(gca,'xticklabel',DateTickLabel)
legend('Canada','France','Germany','Japan','UK','US', ...
        'location','NE')
saveas(gcf,'GlobalIdx102Dif.jpg')
% End of program
% --------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 주가들이 행렬 prices에, 3개월물 Eribor가 벡터 euribor3M에, 주가의 로그수익률들이 행렬 returns에, 날짜가 nalja에 기록된다.

로그수익률이 계산된 날수가 스칼라 nVar에, 기대수익률이 벡터 averReturn에, 로그수익률의 표준편차 즉 변동성이 벡터 sigma에, 그리고 로그수익율의 상관계수가 행렬 corrReturn에 수록된다. 그 결과는 다음과 같다.

```
nVar =
     6

averReturn =
   1.0e-03 *
    0.2214   -0.0713   -0.0582    0.1734   -0.0152   -0.0174

sigma =
    0.0085    0.0147    0.0168    0.0139    0.0114    0.0109

corrReturn =
    1.0000    0.4813    0.5058    0.1854    0.4573    0.6526
    0.4813    1.0000    0.8485    0.2261    0.8575    0.5102
    0.5058    0.8485    1.0000    0.2001    0.7650    0.6136
    0.1854    0.2261    0.2001    1.0000    0.2295    0.1439
    0.4573    0.8575    0.7650    0.2295    1.0000    0.4617
    0.6526    0.5102    0.6136    0.1439    0.4617    1.0000
```

기존의 데이터와 더불어 새로이 생성된 데이터들이 MATLAB데이터파일 Global_Idx102_Data.mat에 저장된다. 또한, 주가들의 시계열산점도인 그림 5.5.2와 로그수익률의 시계열산점도인 그림 5.5.3이 출력된다. ■

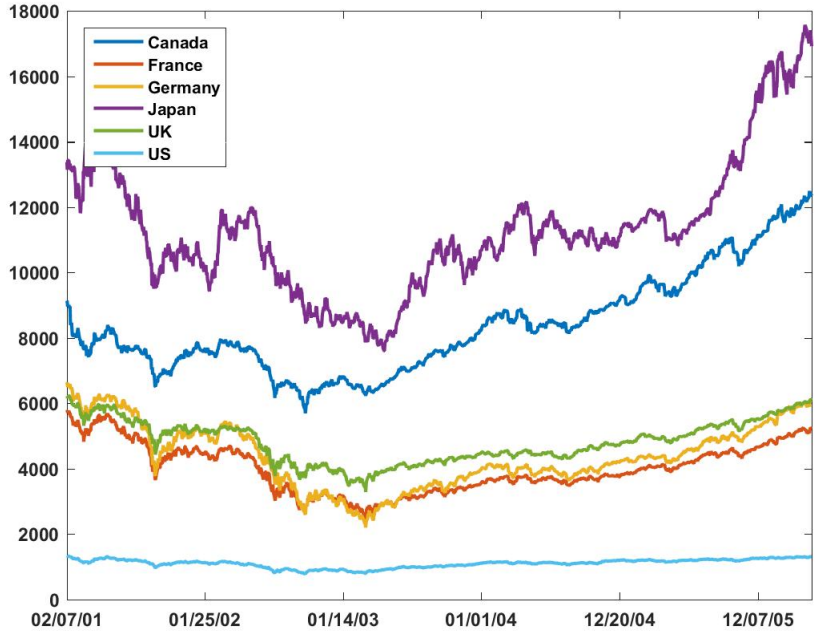


그림 5.5.2. Data_Global_Idx102.mat에 포함된 주가지수들의 시계열산점도

MATLAB함수 sde.m을 사용해서, 다변량 시장모형으로부터 표본경로를 발생시켜보자.

예제 5.5.13 MATLAB함수 sde.m을 사용해서 데이터세트 Global_Idx102_Data.mat의 대형주지수들에 부합되는 표본경로를 발생시키기로 하기 위해서, 다음 MATLAB프로그램 ExampleSDE101.m을 실행해보자.

```
% ------------------------------------------------------------------------
%  Filename: ExampleSDE101.m
%  Simulating Equity Price Example 1
%  Based on Chapter 8 of Econmtrics Toolbox Manual
%-------------------------------------------------------------------------
clear all, close all, clc
load Global_Idx102_Data
% Initial values
t = 0;
X = 100*ones(nVar,1)
% SDE modeling
F = @(t,X) diag(averReturn)*X
G = @(t,X) diag(X)*diag(sigma)
SDE101 = sde(F,G,'Correlation',corrReturn,'StartState',X,'StartTime',t)
% Simulating equity markets
```

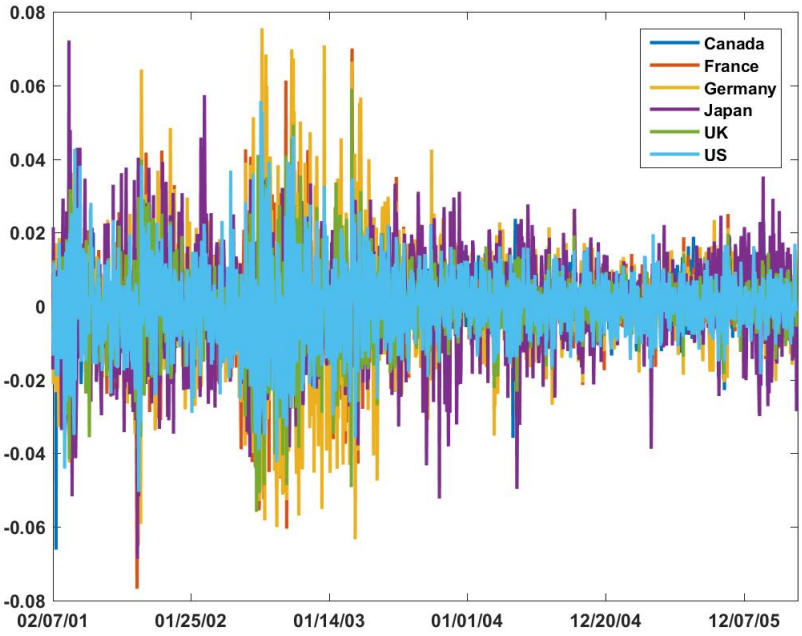


그림 5.5.3. Data_Global_Idx102.mat에 포함된 로그수익률들의 시계열산점도

```
nPeriods =249
dt = 1
rng(5489,'twister')
[S,T] = SDE101.simulate(nPeriods,'DeltaTime',dt);
whos S T
plot(T,S,'linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('Trading Day'), ylabel('Price')
legend('Canada','France','Germany','Japan','UK','US', ...
       'location','NW')
saveas(gcf,'ExampleSDE101.jpg')
% End of program
% --------------------------------------------------------------------------
```

MATLAB함수 sde.m을 사용해서 표본경로를 생성하기 위해서, 반드시 추세계수함수 F와 확산계수함수 G를 지정해야 한다. 이 sde명령문에서 추세계수함수 F는 다음과 같다.

$$F\left(t,\boldsymbol{x}_t\right)=diag\left(\bar{r}_1,\bar{r}_2,\bar{r}_3,\bar{r}_4,\bar{r}_5,\bar{r}_6\right)\boldsymbol{x}_t \tag{1}$$

여기서 $x_{i,t}$는 제i번째 나라의 시점 t에서 주가이고, $\boldsymbol{x}_t=[x_{1,t},x_{2,t},x_{3,t},x_{4,t},x_{5,t},x_{6,t}]^t$이며, $\bar{r}_i$는 제i번째 나라의 평균주가수익률이다. 또한, 확산계수함수 G는 다음과 같다.

$$G\left(t,\boldsymbol{x}_t\right)=diag\left(x_{1,t}\sigma_1,x_{2,t}\sigma_2,x_{3,t}\sigma_3,x_{4,t}\sigma_4,x_{5,t}\sigma_5,x_{6,t}\sigma_6\right) \tag{2}$$

여기서 σ_i는 제i번째 나라의 주가변동성이다. 또한, 확률미분방정식은 다음과 같다.

$$d\boldsymbol{x}_t = F(t, \boldsymbol{x}_t)\, dt + G(t, \boldsymbol{x}_t)\, d\boldsymbol{W}_t \tag{3}$$

모수 StartTime과 모수 StartState에 각각 t와 X를 지정했으므로, 표본경로의 초기시점 0에서 초기값을 100으로 한다. 이는 각 나라의 초기시점에서 주가를 100으로 고정시키는 것이다. 모수 StartTime의 디폴트는 0이다. 모수 correlation에 corrReturn을 할당했으므로, 다변량 Brown운동 $\{\boldsymbol{W}_t\}$의 상관계수행렬은 로그수익율의 상관계수들의 행렬 corrReturn이다. 모수 simulation이 지정되지 않았으므로, 시뮬레이션방법으로 디폴트인 simEuler가 사용된다. 이 sde명령문을 실행하면, 확률미분방정식 (3)의 모형이 SDE101에 수록된다. 그 결과는 다음과 같다.

```
F =
    @(t,X)diag(averReturn)*X

G =
    @(t,X)diag(X)*diag(sigma)

SDE1 =
   Class SDE: Stochastic Differential Equation
   -------------------------------------------
     Dimensions: State = 6, Brownian = 6
   -------------------------------------------
      StartTime: 0
     StartState: 100 (6x1 double array)
    Correlation: 6x6 double array
          Drift: drift rate function F(t,X(t))
      Diffusion: diffusion rate function G(t,X(t))
     Simulation: simulation method/function simByEuler
```

식 (3)의 확률미분방정식에서 표본경로를 발생하기 위해서는 다음과 같은 MATLAB 명령문을 실행한다.

```
>> [S,T] = SDE1.simulate(nPeriods,'DeltaTime',dt)
```

이 MATLAB명령문에서 nPeriods는 각 표본경로에서 관찰값들을 발생시킬 시점들의 개수이고 모수 DeltaTime은 시점들 사이의 길이, 즉 시간간격이다. 유의할 점은 nPeriods에는 초기시점이 포함되지 않는다는 것이다. 발생된 표본경로들을 행렬 S에 저장하고 시점들을 벡터 T에 저장한다. 이 MATLAB명령문을 실행하면, 다음과 같은 결과를 출력한다. 또한, 생성된 표본경로를 그린 것이 그림 5.5.4이다.

```
nPeriods =
```

```
    249

dt =
     1

  Name         Size              Bytes  Class     Attributes

  S          250x6              12000  double
  T          250x1               2000  double
```

■

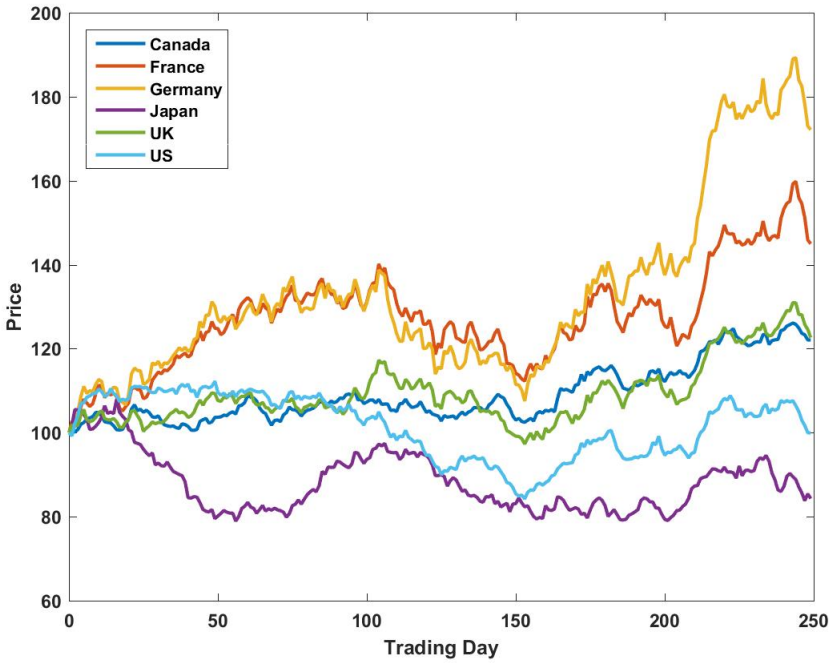


그림 5.5.4. 표본경로의 시계열산점도

예제 5.5.14 다음 MATLAB프로그램 ExampleSDE102.m를 수행해도 MATLAB프로그램 ExampleSDE101.m과 동일한 결과를 얻는다.

```
% --------------------------------------------------------------------
%  Filename: ExampleSDE102.m
%  Simulating Equity Price Example 2
%  Produce the Same Result as ExampleSDE101.m
%--------------------------------------------------------------------
clear all, close all, clc
load Global_Idx102_Data
% Initial values
t = 0;
X = 100*ones(nVar,1)
% SDE modeling
F = @(t,X) diag(averReturn)*X
```

```
G = @(t,X) diag(X)*diag(sigma)
SDE102 = sde(F,G,'correlation',corrReturn,'StartState',X)
% Simulating equity markets
nPeriods =249
dt = 1
strm1 = RandStream('mt19937ar','Seed',5489)
        % mt19937ar = Mersenne Twister
RandStream.setGlobalStream(strm1)
[S,T] = SDE102.simByEuler(nPeriods,'DeltaTime',dt);
whos S T
plot(T,S,'linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('Trading Day'), ylabel('Price')
legend('Canada','France','Germany','Japan','UK','US', ...
       'location','NW')
saveas(gcf,'ExampleSDE102.jpg')
% End of program
% ------------------------------------------------------------------------
```

■

MATLAB함수 sde.m를 사용해서 표본경로들을 반복해서 생성하기 위해서는 옵션 nTrials를 사용한다.

예제 5.5.15 MATLAB함수 sde.m을 사용해서 데이터세트 Global_Idx102_Data.mat의 대형주지수들에 부합되는 복수의 표본경로들을 발생시키기로 하기 위해서, 다음 MATLAB 프로그램 ExampleSDE103.m을 실행해보자.

```
% ------------------------------------------------------------------------
%  Filename: ExampleSDE103.m
%  Simulating Equity Price Example 3
%  Multiple Samplepaths
%-------------------------------------------------------------------------
clear all, close all, clc
load Global_Idx102_Data
% Initial values
t = 0;
X = 100*ones(nVar,1)
% SDE modeling
F = @(t,X) diag(averReturn)*X
G = @(t,X) diag(X)*diag(sigma)
SDE101 = sde(F,G,'correlation',corrReturn,'StartState',X)
% Simulating equity markets
nPeriods =249
dt = 1
strm1 = RandStream('mt19937ar','Seed',5489) % mt19937ar = Mersenne Twister
RandStream.setGlobalStream(strm1)
[S,T] = SDE101.simByEuler(nPeriods,'DeltaTime',dt,'nTrials',100);
whos S T
plot(T,S(:,:,1),'linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('Trading Day'), ylabel('Price')
legend('Canada','France','Germany','Japan','UK','US', ...
       'location','NW')
```

```
saveas(gcf,'ExampleSDE103a.jpg')
figure
plot(T,S(:,:,100),'linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('Trading Day'), ylabel('Price')
legend('Canada','France','Germany','Japan','UK','US', ...
       'location','NW')
saveas(gcf,'ExampleSDE103b.jpg')
% End of program
% ------------------------------------------------------------------------
```

예제 5.5.13의 확률미분방정식 (3)에서 복수의 표본경로들을 발생하기 위해서 다음 MATLAB명령문을 실행하자.

```
>> [S,T] = SDE101.simByEuler(nPeriods,'DeltaTime',dt',nTrials',100)
```

이 MATLAB명령문에서는 시뮬레이션방법을 지정하기 위해서 함수 simulate.m이 아닌 함수 simByEuler.m을 사용하였다. 그러나, simByEuler가 디폴트이므로 이 경우에는 simulate를 사용하는 것과 같다. 모수 nTrials는 각 국가별로 생성할 표본경로들의 개수이다. 여기서는 모수 nTrials에 100을 할당했으므로, 이 MATLAB명령문을 실행하면 각 국가별로 표본경로를 100개 씩 생성한다. 따라서, 표본경로들을 수록한 행렬 S의 차원은 $[nPeriods + 1] \times NVARS \times nTrials$이다. 다음 결과물에서 이를 확인할 수 있다.

```
  Name         Size                Bytes  Class     Attributes
    S          250x6x10           120000  double
    T          250x1                2000  double
```

행렬 S의 세 번째 원소가 표본경로의 번호를 나타낸다. 따라서, 이 세 번째 원소가 1인 행렬 S의 차원은 250×6이다. 즉, 행렬 S(:,:,1)의 시계열산점도는 그림 5.5.4과 동일하다. ■

MATLAB함수 sdeddo.m을 사용해서, 다변량 시장모형으로부터 표본경로를 발생시켜 보자.

예제 5.5.16 MATLAB함수 sdeddo.m을 사용해서 데이터세트 Global_Idx102_Data.mat의 대형주지수들에 부합되는 표본경로를 발생시키기로 하기 위해서, 다음 MATLAB프로그램 ExampleSDEDDO1.m을 실행해보자.

```
% ------------------------------------------------------------------------
%  Filename: ExampleSDEDDO101.m
%  Simulating Equity Price using SDEDDO Object 1
%-------------------------------------------------------------------------
```

```
clear all, close all, clc
load Global_Idx2_Data
% Initial values
t = 0;
X = 100*ones(nVar,1);
% SDEDDO modeling
F = drift(zeros(nVar,1),diag(averReturn))
G = diffusion(ones(nVar,1),diag(sigma))
SDEDDO101 = sdeddo(F,G,'Correlation',corrReturn,'StartState',100)
% Simulating equity markets
nPeriods =249
dt = 1
rng(5489,'twister')
[S,T] = SDEDDO101.simulate(nPeriods,'DeltaTime',dt);
whos S T
plot(T,S,'linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('Trading Day'), ylabel('Price')
legend('Canada','France','Germany','Japan','UK','US', ...
       'location','NW')
saveas(gcf,'ExampleSDEDDO101.jpg')
% End of program
% --------------------------------------------------------------------------
```

MATLAB함수 sde.m 보다는 MATLAB함수 sdeddo.m에서 추세계수함수와 확산계수함수에 대해 좀 더 많은 정보를 요구한다. 따라서, MATLAB함수 sdeddo.m에서는 입력변수의 차원을 좀 더 쉽게 다룰 수 있다. 예를 들어, 예제 5.5.13의 sde명령문에서는 모수 StartState에 정확한 차원의 벡터 X를 할당했으나, 이 예제의 sdeddo명령문에서는 모수 StartState에 상수 100을 할당했다. 이 sdeddo명령문을 실행하면, 다변량 시장모형이 SDEDOO101에 수록된다. 그 결과는 다음과 같다.

```
F =
   Class DRIFT: Drift Rate Specification
   -------------------------------------
      Rate: drift rate function F(t,X(t))
         A: 6x1 double array
         B: 6x6 diagonal double array

G =
   Class DIFFUSION: Diffusion Rate Specification
   ---------------------------------------------
       Rate: diffusion rate function G(t,X(t))
      Alpha: 6x1 double array
      Sigma: 6x6 diagonal double array

SDEDDO101 =
   Class SDEDDO: SDE from Drift and Diffusion Objects
   --------------------------------------------------
     Dimensions: State = 6, Brownian = 6
   --------------------------------------------------
      StartTime: 0
     StartState: 100 (6x1 double array)
    Correlation: 6x6 double array
          Drift: drift rate function F(t,X(t))
```

```
      Diffusion: diffusion rate function G(t,X(t))
     Simulation: simulation method/function simByEuler
              A: 6x1 double array
              B: 6x6 diagonal double array
          Alpha: 6x1 double array
          Sigma: 6x6 diagonal double array

nPeriods =
   249

dt =
     1

  Name         Size              Bytes  Class     Attributes

  S          250x6              12000  double
  T          250x1               2000  double
```

MATLAB프로그램 ExampleSDEDDO1.m를 수행하면, MATLAB프로그램 ExampleSDE101.m과 동일한 표본경로를 얻는다. 즉, 그림 5.5.4과 동일한 그림을 얻는다. ■

예제 5.5.17 MATLAB함수 gbm.m을 사용해서 데이터세트 Global_Idx102_Data.mat의 대형주지수들에 부합되는 표본경로들을 발생시키기로 하기 위해서, 다음 MATLAB프로그램 SDEtest101.m을 실행해보자.

```
% ------------------------------------------------------------------------
%  Filename: SDEtest101.m
%  Generating Random Numbers from Multidimensional Market Model 1
%  Programmed by ByoungSeon CHOI
%-------------------------------------------------------------------------
clear all, close all, clc
load Global_Idx102_Data
% Initial values
t = 0;  X = 100*ones(nVar,1);
nPeriods = 249,  dt = 1
% SDEDDO modeling with simByEuler
F = drift(zeros(nVar,1),diag(averReturn))
G = diffusion(ones(nVar,1),diag(sigma))
SDEDD1 = sdeddo(F,G,'Correlation',corrReturn,'StartState',100)
rng(5489,'twister')
[S,T] = SDEDD1.simulate(nPeriods,'DeltaTime',dt);
subplot(2,2,1)
plot(T,S,'linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 250])
title('\bf SDEDDO w/ simByEuler')
% SDELD modeling with simByEuler
SDELD101 = sdeld(zeros(nVar,1),diag(averReturn), ...
                 ones(nVar,1),diag(sigma), ...
                 'Correlation',corrReturn,'StartState',X)
rng(5489,'twister')
[S,T] = SDELD101.simulate(nPeriods,'DeltaTime',dt);
subplot(2,2,2)
plot(T,S,'linewidth',2)
```

```
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 250])
title('\bf SDELD w/ simByEuler')
% CEV modeling with simByEuler
CEV1 = cev(diag(averReturn),ones(nVar,1),diag(sigma), ...
           'Correlation',corrReturn,'StartState',X)
rng(5489,'twister')
[S,T] = CEV1.simulate(nPeriods,'DeltaTime',dt);
subplot(2,2,3)
plot(T,S,'linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 250])
title('\bf CEV w/ simByEuler')
% GBM modeling with simByEuler
GBM1 = gbm(diag(averReturn),diag(sigma), ...
           'Correlation',corrReturn,'StartState',X)
rng(5489,'twister')
[S,T] = GBM1.simulate(nPeriods,'DeltaTime',dt);
subplot(2,2,4)
plot(T,S,'linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 250])
title('\bf GBM w/ simByEuler')
saveas(gcf,'SDEtest101.jpg')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB함수을 실행하면 그림 5.5.5가 그려진다. 5.5.5의 좌측상단에는 MATLAB함수 sdeddo.m에 의해서 발생된 표본경로가, 우측상단에는 MATLAB함수 sdeld.m에 의해서 발생된 표본경로가, 좌측하단에는 MATLAB함수 cev.m에 의해서 발생된 표본경로가, 우측하단에는 MATLAB함수 gbm.m에 의해서 발생된 표본경로가 수록되어 있다. 그림 5.5.5에서 알 수 있듯이, MATLAB함수 gbm.m에 의해서 발생된 표본경로는 다른 함수들에 의해서 발생된 표본경로와는 다르다. ■

기하Brown운동의 해석해가 알려져 있으므로, MATLAB함수 gbm.m에서는 옵션 Simulation에 simBySolution을 할당함으로써 이 해석해를 사용할 수 있다.

예제 5.5.18 MATLAB함수 gbm.m에서 시뮬레이션방법으로 simBySolution을 사용해서 데이터세트 Global_Idx102_Data.mat의 대형주지수들에 부합되는 표본경로들을 발생시키기로 하기 위해, 다음 MATLAB프로그램 SDEtest102.m을 실행해보자.

```
% ------------------------------------------------------------------------
%  Filename: SDEtest102.m
%  Generating Random Numbers from Multidimensional Market Model 2
%  Programmed by ByoungSeon CHOI
%-------------------------------------------------------------------------
clear all, close all, clc
load Global_Idx2_Data
% Initial values
t = 0; X = 100*ones(nVar,1);
nPeriods = 249, dt = 1
```

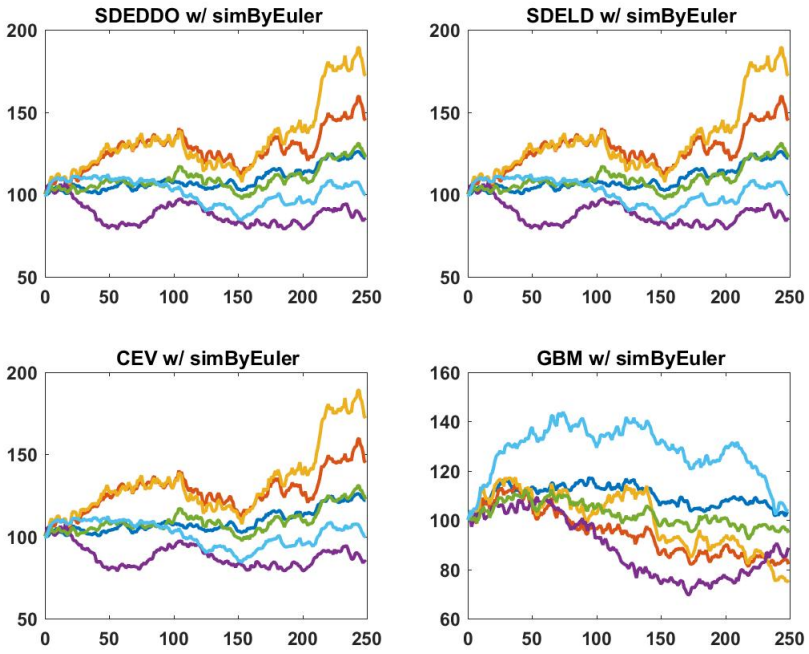


그림 5.5.5. 다양한 MATLAB함수들에 의해서 생성된 표본경로들

```
% SDE modeling with simByEuler
F = @(t,X) diag(averReturn)*X
G = @(t,X) diag(X)*diag(sigma)
SDE1 = sde(F,G,'Correlation',corrReturn,'StartState',X,'StartTime',t)
rng(5489,'twister')
[S,T] = SDE1.simulate(nPeriods,'DeltaTime',dt);
subplot(2,2,1)
plot(T,S,'linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 250])
title('\bf SDE w/ simByEuler')
% SDELD modeling with simByEuler
SDELD1 = sdeld(zeros(nVar,1),diag(averReturn), ...
                 ones(nVar,1),diag(sigma), ...
                 'Correlation',corrReturn,'StartState',X)
rng(5489,'twister')
[S,T] = SDELD1.simulate(nPeriods,'DeltaTime',dt);
subplot(2,2,2)
plot(T,S,'linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 250])
title('\bf SDELD w/ simByEuler')
% GBM modeling w/ simByEuler
GBM1 = gbm(diag(averReturn),diag(sigma), ...
           'Correlation',corrReturn,'StartState',X)
rng(5489,'twister')
[S,T] = GBM1.simByEuler(nPeriods,'DeltaTime',dt);
subplot(2,2,3)
plot(T,S,'linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 250])
title('\bf GBM w/ simByEuler')
% GBM modeling w/ simBySolution
GBM1 = gbm(diag(averReturn),diag(sigma), ...
           'Correlation',corrReturn,'StartState',X)
rng(5489,'twister')
```

```
[S,T] = GBM1.simBySolution(nPeriods,'DeltaTime',dt);
subplot(2,2,4)
plot(T,S,'linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 250])
title('\bf GBM w/ simBySolution')
saveas(gcf,'SDEtest102.jpg')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB함수을 실행하면 그림 5.5.6가 그려진다. 그림 5.5.6의 좌측상단에는 MATLAB함수 sde.m의 옵션 Simulation에 디폴트인 simByEuler를 적용해서 발생시킨 표본경로가, 우측상단에는 MATLAB함수 sdeld.m의 옵션 Simulation에 디폴트인 simByEuler를 적용해서 발생시킨 표본경로가, 좌측하단에는 MATLAB함수 gbm.m의 옵션 Simulation에 디폴트인 simByEuler를 적용해서 발생시킨 표본경로가, 우측하단에는 MATLAB함수 gbm.m의 옵션 Simulation에 디폴트인 simBySolution를 적용해서 발생시킨 표본경로가 수록되어 있다. 이 그림에서 알 수 있듯이, MATLAB함수 gbm.m의 옵션 Simulation에 simBySolution을 적용해서 발생시킨 표본경로는 MATLAB함수들 sde.m, sdeddo.m, sdeld.m, cev.m의 옵션 Simulation에 디폴트인 simByEuler를 적용해서 발생시킨 표본경로들과 아주 비슷하다. ■

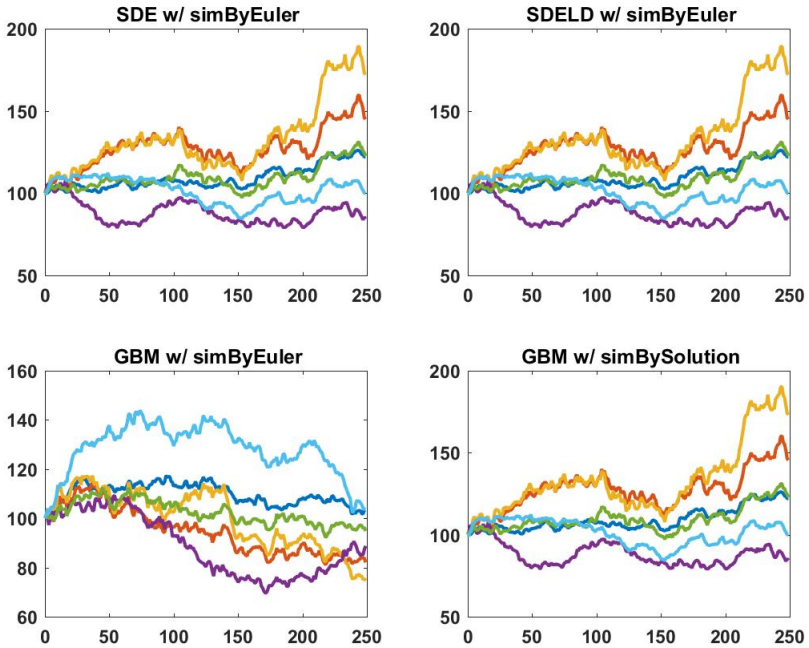


그림 5.5.6. MATLAB함수 gbm.m의 simBySolution에 의해서 발생된 표본경로

아마도 독자들도 예제 5.5.17과 예제 5.5.17에서 무언가 이상하다는 느낌을 갖았을 것이다.

MATLAB함수들 sde.m, sdeddo.m, sdeld.m 그리고 cev.m의 simByEuler에 의한 표본경로들은 동일하고, 또한 gbm.m의 simBySolution에 의한 표본경로도 이들과 같다. 그러나, MATLAB함수 gbm.m의 simByEuler에 의한 표본경로는 이들과 다르다. 함수 gbm.m이 다른 함수들 sde.m, sdeddo.m, sdeld.m, 그리고 cev.m의 특수한 함수라는 것을 고려하면, 이 결과는 매우 이상한 것이다. 본저자는 그림 5.5.5와 그림 5.5.6을 미국 MathWorks사에 보내 이 결과에 대해 질문하였고, 다음과 같은 답을 받았다.

> "I looked at the JPG results, and believe I understand the issue.
> What the user is observing is the result of an overloaded "simByEuler" method specific to the GBM model. In certain situations (e.g., diagonal drift, static volatility), the GBM Euler simulation calls a method which re-factors the underlying equation, and instead of directly simulating the states X(t), which usually represent prices for a GBM model, the overloaded method simulates a sequence of identical returns. These returns are then eventually compounded to produce the corresponding prices. This allows the overloaded GBM simByEuler method to boldorize across time, which is to say that there is no explicit loop over time at all. This boldorization confers the specialized method, when applicable, a huge runtime performance improvement. Any Monte Carlo simulation will generate identical results on a path-by-path basis only if the same random numbers are applied in exactly the same order, but here they are not.
> The reason why SDE, SDEDDO, SDELD, and CEV generate the same paths when started from the same seed is that they all call the same top-level SDE simByEuler method. However, the GBM model calls a totally different method.
> Simply put, the results are not the same because a different code path is being taken, and this is not a bug but rather an explicit choice. "

예제 5.5.19 MATLAB함수 GBM.m의 시뮬레이션방법들 simByEuler와 simBySolution의 차이를 좀 더 살펴보기 위해서, 다음 MATLAB프로그램 ExampleGBM101.m을 실행해보자.

```
% ------------------------------------------------------------------------
%  Filename: ExampleGBM101.m
%  Simulating Equity Price using GBM Object 1
%  GBM Simulation Methods; simulate and symBySolution
%------------------------------------------------------------------------
clear all, close all, clc
load Global_Idx102_Data
% Initial values
t = 0; X = 100*ones(nVar,1);
```

```
nPeriods = 249, dt = 1
% GBM modeling
GBM1 = gbm(diag(averReturn),diag(sigma), ...
           'correlation',corrReturn,'StartState',X)
% Simulating equity markets by symByEuler
nPeriods = 249
dt = 1
rng(5489,'twister')
[S,T] = GBM1.simByEuler(nPeriods,'DeltaTime',dt,'nTrials',10000);
% Simulating equity markets by symBySolution
rng(5489,'twister')
[U,T] = GBM1.simBySolution(nPeriods,'DeltaTime',dt,'nTrials',10);
% Plotting
[ Freq Price ] = hist(squeeze(S(end,1,:)),30);
bar(Price,Freq,'w')
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('Price'), ylabel('Frequency')
saveas(gcf,'ExampleGBM101a.jpg')
figure
plot(T,S(:,:,1)-U(:,:,1),'linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
ylabel('\bf Price Difference')
saveas(gcf,'ExampleGBM101b.jpg')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램에서 행렬 S는 MATLAB함수 gbm.m에서 simByEuler방법을 사용해서 발생시킨 10,000개 표본경로들을 저장한 것이다. 따라서, 행렬 S의 차원은 $250\times6\times10000$이다. 또한, 행렬 U는 MATLAB함수 gbm.m에서 simBySolution방법을 사용해서 발생시킨 10개 표본경로들을 저장한 것이다. 따라서, 행렬 U의 차원은 $250\times6\times10$이다.

이 MATLAB프로그램에서 사용한 MATLAB함수 hist.m 안에서 사용된 MATLAB함수 squeeze.m는 단차원(singleton dimension)을 제거해서 행렬의 차원을 축소하는 함수이다. 여기서 사용된 squeeze(S(end,1,:))은 S(end,1,:)에서 단차원인 첫 번째 원소와 두 번째 원소의 차원들을 제거해서 벡터로 만든다. 따라서, squeeze(S(end,1,:))은 마지막 거래일에 첫 번째 변수인 캐나다 대형주지수 TSX의 10,000개 시뮬레이션값들을 저장한 $10,000\times1$ 열벡터이다. 이 시뮬레이션값들의 히스토그램이 그림 5.5.7에 수록되어 있다. 그림 5.5.7에서 알 수 있듯이, 캐나다 대형주지수 TSX는 로그정규확률분포를 따른다는 것을 알 수 있다.

시뮬레이션방법들 simByEuler와 simBySolution에 의한 첫 번째 표본경로의 차이를 그린 것이 그림 5.5.8이다. MATLAB Finance Toolbox에 기술된 것과는 달리, 시뮬레이션방법들 simByEuler와 simBySolution에 의한 표본경로들의 차이가 크다는 것을 알 수 있다. ■

주가지수들 사이의 상관관계를 나타내는 방법들을 살펴보기 위해서, 다음 예제를 살펴보자.

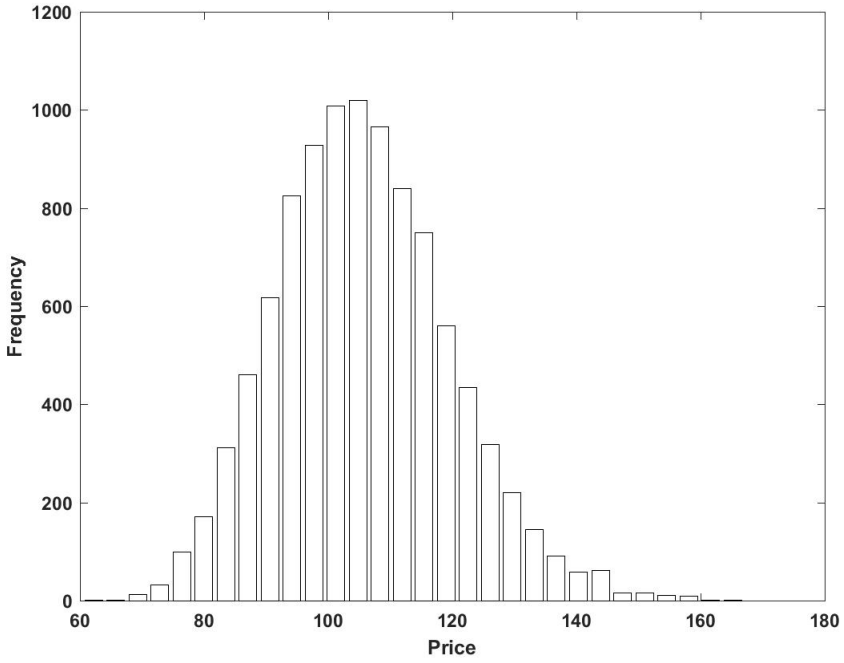


그림 5.5.7. MATLAB함수 gbm.m에 의해서 발생된 관찰값들의 히스토그램

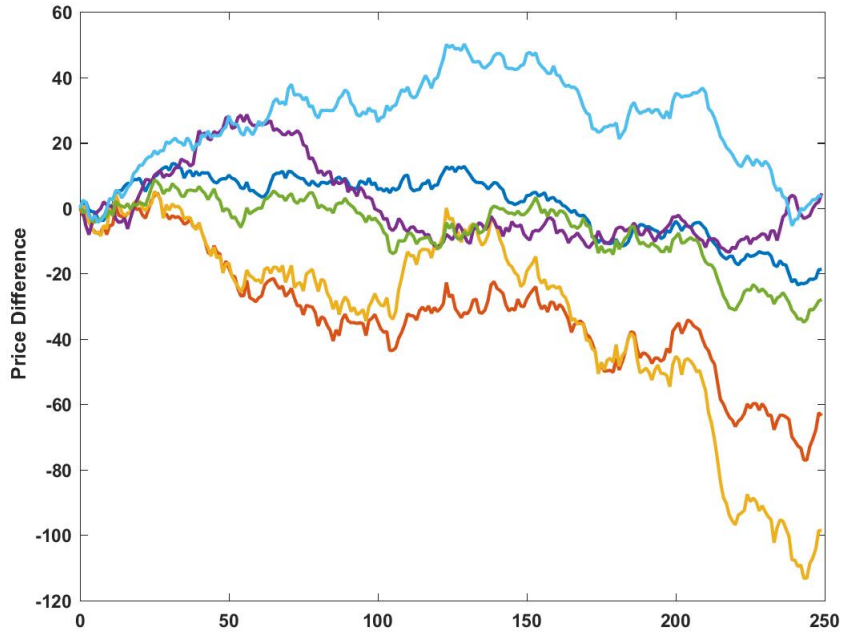


그림 5.5.8. 시뮬레이션방법에 의한 표본경로들의 차이

예제 5.5.20 다음과 같은 다변량 시장모형을 살펴보자.

$$d\boldsymbol{x}_t = \mu \boldsymbol{x}_t dt + D\left(\boldsymbol{x}_t\right) \sigma d\boldsymbol{W}_t \tag{1}$$

여기서 μ는 기대수익률을 나타내는 대각행렬이고, $D(\boldsymbol{x}_t)$는 제i번째 대각원소가 $\boldsymbol{x}_t$의 제i번째 원소 $x_{i,t}$인 대각행렬이고, σ는 제i번째 주가수익률의 표준편차인 대각행렬이다. 만약 $d\boldsymbol{W}_t$의 분산공분산행렬을 Pdt 라고 하면, $\sigma d\boldsymbol{W}_t$의 분산공분산행렬은 $\sigma P\sigma^t dt$이다. 즉, 주가지수수익률들의 분산공분산행렬은 $\sigma P\sigma^t$이다. Cholesky분해를 해서, 다음 식을 만족하는 상삼각행렬을 구하자.

$$\sigma P\sigma^t = T^t T \tag{2}$$

다음 확률미분방정식을 살펴보자.

$$d\boldsymbol{x}_t = \mu \boldsymbol{x}_t dt + D(\boldsymbol{x}_t)\, T^t d\widetilde{\boldsymbol{W}}_t \tag{3}$$

여기서 $d\widetilde{\boldsymbol{W}}_t$의 분산공분산행렬은 Idt이다. 즉, $\left\{\widetilde{\boldsymbol{W}}_t\right\}$는 표준다변량Brown운동이다.

확률미분방정식들 (1)과 (3)이 동일한 표본경로를 발생시킴을 확인하기 위해서, 다음 MATLAB프로그램 SDEcorrelation101.m을 실행해보자.

```
% ------------------------------------------------------------------------
%  Filename: SDEcorrelation101.m
%  Representing Correlations when Simulating Equity Prices
%-------------------------------------------------------------------------
clear all, close all, clc
load Global_Idx2_Data
% Initial values
t = 0; X = 100*ones(nVar,1);
% SDELD modeling 1
SDELD1 = sdeld(zeros(nVar,1),diag(averReturn),ones(nVar,1),diag(sigma), ...
               'correlation',corrReturn,'StartState',X)
rng(22814,'twister')
[S1,T] = SDELD1.simByEuler(1000);
% SDE modeling 2
covReturn = cov(returns)
Sigma05 = cholcov(covReturn)'    % Sigma05 should be lower-triangular
SDELD2 = sdeld(zeros(nVar,1),diag(averReturn),ones(nVar,1),Sigma05, ...
               'StartState',X)
rng(22814,'twister')
[S2,T] = SDELD2.simByEuler(1000);
% Plotting
plot(T,S1-S2,'linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
xlabel('Tradng Day'), ylabel('Difference')
saveas(gcf,'SDEcorrelation101.jpg')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB함수을 실행하면 그림 5.5.9이 그려진다. 이 그래프는 확률미분방정식 (1)로부터 발생한 표본경로에서 확률미분방정식 (3)으로부터 발생한 표본경로를 뺀 것이다. 이 그래프들의 절대값은 5×10^{-13} 보다 작다. 따라서, 이 두 방법들이 동일한 표본경로들을

발생시킴을 알 수 있다. ■

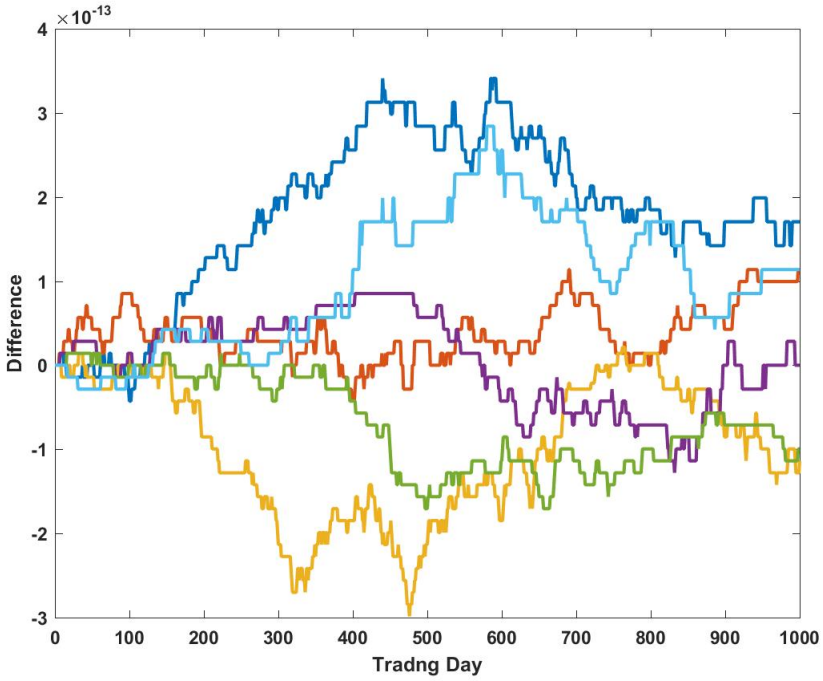


그림 5.5.9. 상관관계를 나타내는 방법을 달리한 표본경로들의 차이

MATLAB함수 ts2func.m을 사용하면, 모수가 상수가 아닌 시간의 함수인 경우에도 SDE류의 MATLAB함수를 사용해서 확률미분방정식에서 표본경로를 생성할 수 있다.

예제 5.5.21 SDE류의 MATLAB함수에서는 보편적 인터페이스(common interface)를 사용해서 모수를 지정할 수 있다. 이러한 기능 때문에, SDE류의 MATLAB함수에서 시간종속적(time-dependent 또는 dynamic) 모수를 사용할 수 있다. 따라서, SDE류의 MATLAB함수를 사용해서 복잡한 비선형 확률모형으로부터 표본경로를 생성할 수 있다. 이러한 목적으로 사용되는 MATLAB함수가 ts2func.m이다.

무위험이자율이 시간종속적인 기하Brown운동모형을 사용해서 Deutsche Borse AG German Stock Index인 DAX지수의 표본경로를 생성하기 위해서, 다음 MATLAB프로그램 SDEdynamic101.m을 실행해보자.

```
% ------------------------------------------------------------------------
%  Filename: SDEdynamic101.m
%  Representing Dynamic Behavior of Market Parameters w/ ts2func.m
%-------------------------------------------------------------------------
clear all, close all, clc
% Dataset Step
load Data_GlobalIdx2
```

```
dt = 1/250
returns = price2ret(Dataset.DAX);
sigma = std(returns)*sqrt(250)
nalja = datestr(dates);
yields = 360*log(1+ Dataset.EB3M);
nPeriods = length(yields)   % Number of simulatated observations
% Constant Risk-free Interest Rate Step
rng(5489,'twister')
GBM1 = gbm(mean(yields),diag(sigma),'StartState',100)
[S1,T] = GBM1.simulate(nPeriods,'DeltaTime',dt);
% Dynamic Risk-free Interest Rate Step
r = ts2func(yields,'Times',(0:nPeriods-1)');
r1 = r(3.5)
r2 = r(3.5,100)
r3 = r(3.5,200)
rng(5489,'twister')
GBM2 = gbm(r,diag(sigma),'StartState',100)
[S2,T] = GBM2.simulate(nPeriods,'DeltaTime',dt);
% Plotting Step
subplot(2,1,1)
plot(dates,100*yields,'linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[730889 732791])
DateTick = dates(1:252:end);
DateTickLabel = datestr(DateTick,2);
set(gca,'xtick',DateTick,'xticklabel',DateTickLabel,'ylim',[1 5])
title('\bf Eribor 3M')
xlabel('Date'), ylabel('Annual Rate (%)')
subplot(2,1,2)
plot(T,S1,'r-',T,S2,'k-.','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 5.457],'ylim',[40 150])
title('\bf Constant vs Dynamic Rate of Return: DAX')
xlabel('Year'), ylabel('Index Level')
legend('Constant','Dynamic','location','SW')
saveas(gcf,'SDEdynamic101.jpg')
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램은 다음 확률미분방정식으로부터 표본경로를 발생시키기 위한 것이다.

$$dS_t = r(t)S_t dt + \sigma S_t dW_t \tag{1}$$

여기서 $r(t)$는 무위험이자율이다.

먼저 Dataset스텝을 살펴보자. 이 데이터세트 Data_GlobalIdx102에는 각 해의 거래일수가 250일로 고정되어 있으므로, 관찰점들 사이의 시간간격을 dt = 1/250으로 하고, DAX 지수의 로그수익률을 벡터 returns에, 이 로그수익률의 연표준편차(annualized standard deviation)를 스칼라 sigma에, 그리고 Euribor 3개월물의 로그수익률을 벡터 yields에 수록한다. 또한, 벡터 yields의 차원을 생성할 표본경로의 길이로 할당한다.

상수인 로그수익률 yields의 표본평균 $\bar{r}$를 무위험이자율로 하는 확률미분방정식은 다음과

같다.

$$dS_t = \bar{r} S_t dt + \sigma S_t dW_t \tag{2}$$

이에 해당하는 GBM모형인 GBM1은 다음과 같다.

```
GBM1 =
   Class GBM: Generalized Geometric Brownian Motion
   -------------------------------------------------
     Dimensions: State = 1, Brownian = 1
   -------------------------------------------------
      StartTime: 0
     StartState: 100
    Correlation: 1
          Drift: drift rate function F(t,X(t))
      Diffusion: diffusion rate function G(t,X(t))
     Simulation: simulation method/function simByEuler
         Return: 0.0278117
          Sigma: 0.266238
```

Euribor 3개월물의 로그수익률 yields를 무위험이자율로 하는 GBM모형을 구축하기로 하자. 이 로그수익률은 시계열이므로, 다음과 같은 MATLAB함수 ts2func.m을 사용해서 함수로 변형하기로 하자.

```
>> r = ts2func(yields,'Times',(0:nPeriods-1)')
```

이 MATLAB명령문은 시계열데이터 yields를 함수 r로 바꾼다. MATLAB함수 ts2func.m의 옵션 Times에는 시계열의 관찰시점들을 열벡터로 나열한다. 이 옵션의 디폴트는 시작시점이 0이고 증가단위가 1이며, 차원은 시계열데이터의 관찰점들의 개수와 같다. 이렇게 생성된 함수는 시점 만의 함수이다. 상태변수를 독립변수로 사용할 수는 있으나, 함수값에는 영향을 미치지는 않는다. 즉, 다음 함수값들은 동일하다.

$$r(3.5) = r(3.5, 100) = r(3.5, 200) = 0.0471 \tag{3}$$

시계열인 로그수익률 yields를 무위험이자율로 하는 확률미분방정식은 다음과 같다.

$$dS_t = r(t) S_t dt + \sigma S_t dW_t \tag{4}$$

이에 해당하는 GBM모형인 GBM2는 다음과 같다.

```
GBM2 =
   Class GBM: Generalized Geometric Brownian Motion
   -------------------------------------------------
```

```
  Dimensions: State = 1, Brownian = 1
-----------------------------------------------
   StartTime: 0
  StartState: 100
 Correlation: 1
       Drift: drift rate function F(t,X(t))
   Diffusion: diffusion rate function G(t,X(t))
  Simulation: simulation method/function simByEuler
      Return: function ts2func/boldor2Function
       Sigma: 0.266238
```

MATLAB함수을 실행하면 그림 5.5.10이 그려진다. 이 그림의 상단 그래프는 Euribor 3개월물의 시계열산점도이고, 하단 그래프의 적색 실선은 무위험확률이 상수인 확률미분방정식 (2)로부터 발생한 표본경로의 시계열산점도이고, 흑색 반점선은 무위험확률이 시계열인 확률미분방정식 (4)로부터 발생한 표본경로의 시계열산점도이다. ■

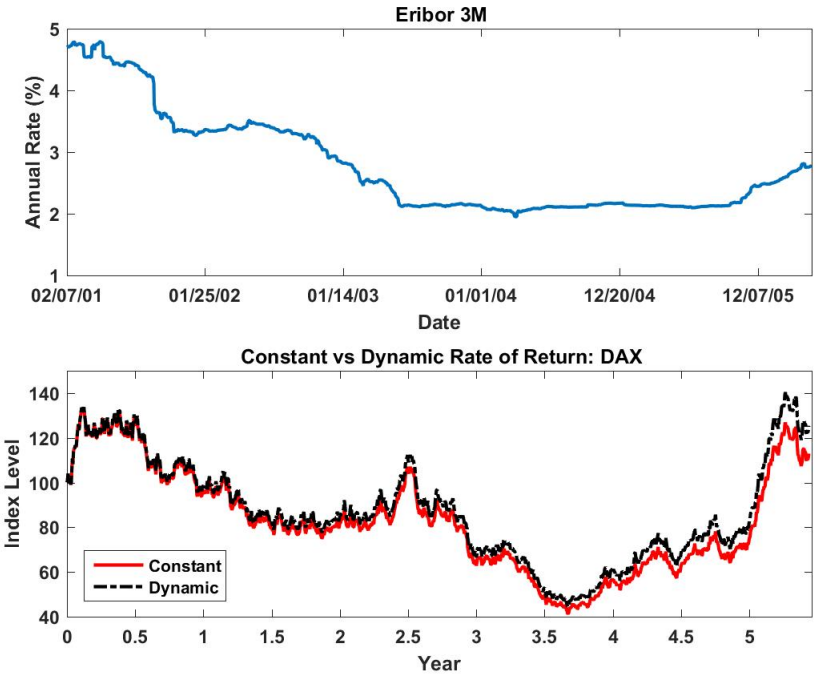


그림 5.5.10. 상수이자율과 동적이자율

SDE류의 MATLAB함수를 사용해서 확률미분방정식에서 표본경로들을 생성하고, 이 표본경로들과 위험중립가치평가식을 이용해서 옵션의 가치를 평가할 수 있다.

예제 5.5.22 위험중립확률측도 하에서 배당이 없는 주가과정 $\{S_t\}$가 다음과 같은 Black-

Scholes확률미분방정식을 따른다고 하자.

$$dS_t = 0.1S_t dt + 0.15S_t dW_t^Q \tag{1}$$

여기서 $\left\{W_t^Q\right\}$는 위험중립확률측도 Q 하에서 Brown운동이다. 따라서, 무위험이자율은 $r = 0.1$이고 S_t의 변동성은 $\sigma = 0.15$이다. 현재시점 $t = 0$에서 원자산이 $S_0 = 18$이고 만기시점 $\tau = 0.25$에서 행사가격이 $K = 15$인 유럽형콜옵션과 유럽형풋옵션의 공정한 가치를 평가하기 위해서, 다음 MATLAB프로그램 SDEoptionPrice101.m을 실행해보자.

```
% ------------------------------------------------------------------------
%  Filename: SDEoptionPrice101.m
%  Pricing European Options using SDE Object
%-------------------------------------------------------------------------
clear all, close all, clc
diary SDEoptionPrice101.txt
% Underlying Model and European Option
K = 110
rate = 0.05
sigma = 0.40
dt = 1/252
nPeriods = 84;
tau = nPeriods*dt
GBM1 = gbm(rate,sigma,'StartState',100)
% End-of-period Processing Function
nTrials = 10000  % Number of Paths
f = Example_BlackScholes(nPeriods,nTrials)
% GBM Objext
rng(5489,'twister')
X = GBM1.simBySolution(nPeriods,'DeltaTime',dt, ...
                       'nTrials',nTrials,'Processes',f.BlackScholes);
% Option Pricing 1
f.CallPrice(K,rate)
f.PutPrice(K,rate)
% Option Pricing 2
CallOption1 = mean(exp(-rate*tau)*max(squeeze(X(end,:,:))-K,0))
PutOption1 = mean(exp(-rate*tau)*max(K-squeeze(X(end,:,:)),0))
save('SDEoptionPrice101','CallOption1','PutOption1')
diary off
% End of program
% ------------------------------------------------------------------------
```

먼저 각 캘린더 달(calender month)의 거래일(trading day)는 21일이라고 가정하자. MATLAB함수 gbm.m을 사용해서 GBM모형 GBM1을 생성한다.

Black-Scholes확률미분방정식 (1)의 해석해가 알려져 있다. 따라서, 이 해석해를 사용하면, 각 표본경로의 중간시점에서 원자산을 생성하지 않고도 행사시점에서 원자산을 생성할 수 있다. 이러한 목적을 위해서 MATLAB함수 Example_BlackScholes.m을 사용할 수 있다. 이 MATLAB함수에 대한 자세한 내용을 알기 위해서는 MATLAB커맨드행에 'help

Example_BlackScholes'를 입력해보라. 이 MATLAB프로그램에서는 다음과 같은 명령문을 사용하였다.

```
>> TSP = Example_BlackScholes(nPeriods,nTrials)
```

즉, 이 MATLAB함수의 입력변수들은 생성되는 표본경로의 길이인 nPeriods와 표본경로들의 개수 nTrials이다. 출력변수(end-of-period processing function) TSP는 행사시점의 표본값들을 저장한 벡터 TSP.BlackScholes, 콜옵션가치를 저장하는 스칼라 TSP.CallPrice, 그리고 풋옵션가치를 저장하는 스칼라 TSP.PutPrice를 포함한다. 즉, 다음과 같은 결과를 출력한다.

```
TSP =
    BlackScholes: @Example_BlackScholes/saveTerminalStockPrice
       CallPrice: @Example_BlackScholes/getCallPrice
        PutPrice: @Example_BlackScholes/getPutPrice
```

GBM모형 GBM1을 사용해서 표본경로들을 생성하기 위해서, 다음과 같은 명령문을 사용한다.

```
>> S = GBM1.simBySolution(nPeriods,'DeltaTime',dt,'nTrials',
  nTrials,'Processes',TSP.BlackScholes);
```

이 MATLAB명령문에서는 해석해를 사용하는 시뮬레이션방법 simBySolution을 사용하였고, 옵션 Processes에 TSP.BlackScholes를 지정함으로써 행사시점에서 표본값들을 TSP.BlackScholes에 저장한다.

유럽형옵션의 공정한 가치를 평가하는 첫 번째 방법으로 저장된 행사시점에서 표본값들 TSP.BlackScholes를 이용하는 것이다. 즉, TSP.CallPrice(K,r)와 TSP.PutPrice(K,r)은 각각 행사가격이 K이고 무위험이자율이 r인 유럽형콜옵션과 유럽형풋옵션의 공정한 가치들이다. 이 예제에서는 유럽형콜옵션가치가 3.3805이고 유럽형풋옵션가치는 0.0009이다.

두 번째 방법은 위험중립가치평가식과 대수법칙을 사용하는 것이다. 그 결과, 예제에서는 유럽형콜옵션가치가 3.3805이고 유럽형풋옵션가치는 0.0009이다.

세 번째 방법은 을 사용하는 것이다. MATLAB함수 blsprice.m을 사용해서 구한 유럽형콜옵션가치는 3.3714이고 유럽형풋옵션가치는 0.0010이다. ■

5.5.3 Brown내삽

SDE류의 MATLAB함수를 사용해서, 확률미분방정식의 표본경로를 Brown내삽(Brownian interpolation)을 할 수 있다. Brown내삽은 결정적(deterministic) 함수가 아닌 확률과정 Brown다리를 사용해서 내삽하는 것이다. Brown내삽을 이해하기 위해서 다음 예제를 살펴보자.

예제 5.5.23 다음과 같은 Brown운동모형에서 표본경로를 생성하고 이 표본경로를 내삽하기로 하자.

$$dx_t = 0.3dt + 0.2dW_t \tag{1}$$

여기서 $\{W_t\}$는 표준Brown운동이다. 이를 실현시키기 위해서, 다음 MATLAB프로그램 SDEinterpolation101.m을 실행하기로 하자.

```
% ------------------------------------------------------------------------
%  Filename: SDEinterpolation101.m
%  Brownian Interpolation w/o Refinement 1
%------------------------------------------------------------------------
clear all, close all, clc, format long
% Generating Sample paths of Brownian motion
BM1 = bm(0.3,0.2);
rng(5489,'twister')
dt = 1/256;
[X,T] = BM1.simulate(256,'DeltaTime',dt);
% Interpolating
tt = ((T(1)+dt/2):dt/2:(T(end)-dt/2))';
x = BM1.interpolate(tt,X,'Times',T);
% Plotting the Sample paths
subplot(2,1,1)
plot(T,X(:,1),'red','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
hold on
plot(tt,x(:,1),'black:','linewidth',2)
legend('Simulated','Interpolated','location','SE')
xlabel('Years'), ylabel('State')
axis([0 1 -0.1 0.4 ])
hold off
subplot(2,1,2)
plot(T,X(:,1),'red-*','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
hold on
plot(tt,x(:,1),'o black','linewidth',2)
legend('Simulated','Interpolated','location','SE')
xlabel('Years'), ylabel('State')
axis([0.4501 0.5001 0.26 0.36])
grid on
hold off
saveas(gcf,'SDEinterpolation101.jpg')
% End of program
% ------------------------------------------------------------------------
```

MATLAB함수 bm.m를 사용해서 생성한 Brown운동모형 MB1으로부터 표본경로를 생성하기 위해서, 다음 명령문을 실행한다.

```
[X,T] = BM1.simulate(256,'DeltaTime',dt);
```

이 명령문을 실행하면, 초기시점 0에서 상태값 0로부터 시작하며 Brown운동모형 (1)로부터 시간간격이 1/256이고 관찰시점들이 256인 표본경로를 생성한다. 즉, 시점을 나타내는 차원이 257×1인 벡터 T는 $[0 : 1/256 : 1]^t$이고 차원이 257×1인 벡터 X는 Brown운동모형 (1)로부터 생성되는 상태벡터이다.

이렇게 생성된 벡터 X를 내삽하기 위해서, 다음 명령문을 실행한다.

```
x = BM1.interpolate(tt,X,'Times',T);
```

여기서 행렬 X와 옵션 Times에 할당된 벡터 T는 주어진 관찰점들을 나타내고, 차원이 511×1인 벡터 tt는 내삽을 하고자 하는 시점을 나타낸다. 즉, 벡터 tt는 $[1/512 : 1/512 : 511/512]^t$이다. 이 MATLAB프로그램을 실행하면, 그림 5.5.11이 그려진다. 이 그림의 상단 그래프에서 적색 실선은 시뮬레이트된 표본경로이고 흑색 점선은 이 표본경로를 내삽한 것이다. 이 그래프를 확대한 것이 하단 그래프이다. 이 하단 그래프에서 적색 실선은 시뮬레이트된 점들을 결정적 함수인 1차식으로 연결한 것이고 흑색 동그라미는 이 표본경로를 확률적으로 내삽한 것이다. ■

SDE류의 MATLAB함수를 사용해서 확률미분방정식의 표본경로를 Brown내삽하는 것은 Brown다리를 사용해서 것임을 확인하기 위해서, 다음 예제를 살펴보자.

예제 5.5.24 예제 5.5.23의 Brown운동모형에서 표본경로를 한 개 생성하고 이 표본경로를 내삽해서 생성한 25,000개 표본경로들의 표본평균과 표본분산을 구하기 위해서, 다음 MATLAB프로그램 SDEinterpolation102.m을 실행해 보자.

```
% ------------------------------------------------------------------------
%  Filename: SDEinterpolation102.m
%  Brownian Interpolation w/o Refinement 2
%-------------------------------------------------------------------------
clear all, close all, clc, format rat
% Generating Sample paths of Brownian motion
BM1 = bm(0.3,0.2);
rng(5489,'twister')
dt = 1/256;
[X,T] = BM1.simulate(256,'DeltaTime',dt);
% Interpolating
```

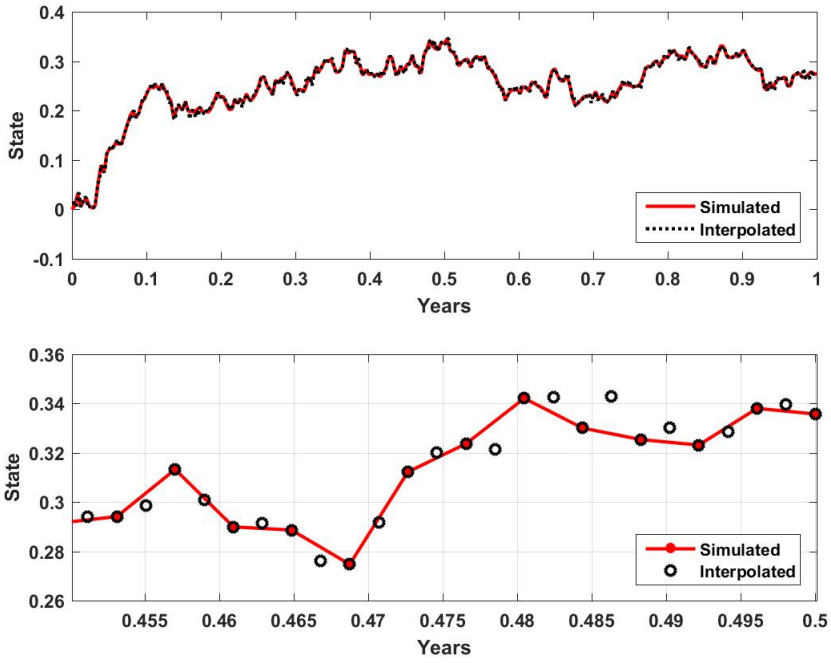


그림 5.5.11. 상수이자율과 동적이자율

```
midNo = 256/2
nTrials = 25000                % Number of tials at each time
ttMid = (T(midNo):dt/8:T(midNo+1))'
format long
XMid = [ X(midNo) X(midNo+1) ]
averInter = zeros(length(ttMid),1);
varInter = zeros(length(ttMid),1);
for ii = 1:length(ttMid)
    tii = ttMid(ii);
    xx = BM1.interpolate(tii(ones(nTrials,1)),X,'Times',T);
    averInter(ii) = mean(xx(:,1));
    varInter(ii) = var(xx(:,1));
end
% Plotting Mean and Variance of Sample paths
subplot(2,1,1)
plot([T(midNo) T(midNo+1)],[X(midNo) X(midNo+1)],'black*','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0.4959 0.5001])
hold on
plot(ttMid,averInter,'redo-','linewidth',2,'Markersize',10)
xlabel('Years'), ylabel('Mean')
grid on
hold off
subplot(2,1,2)
plot([T(midNo) T(midNo+1)],[0 0],'black*','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0.4959 0.5001])
hold on
plot(ttMid,varInter,'redo-','linewidth',2,'Markersize',10)
xlabel('Years'), ylabel('Variance')
grid on
hold off
saveas(gcf,'SDEinterpolation102.jpg')
% End of program
% ------------------------------------------------------------------------
```

예제 5.5.23과 같은 Brown운동모형 MB1으로부터 표본경로를 1개 생성하였다. 이렇게 생성된 상태벡터 X는 다음 식들을 만족한다.

$$X\left(\frac{127}{256}\right) = 0.338,051, \quad X\left(\frac{128}{256}\right) = 0.335,750 \tag{1}$$

다음 MATLAB명령문을 사용해서, 시간구간 $[127/256, 128/256]$을 8개 소구간들로 나눈 차원이 9×1인 분할벡터 ttMid를 구할 수 있다.

```
>> ttMid = (T(midNo):dt/8:T(midNo+1))'
```

앞에서 구한 표본경로를 이 분할에서 내삽하기 위해서, 다음 MATLAB명령문을 실행한다.

```
>> xx = BM1.interpolate(tii(ones(nTrials,1),X,'Times',T);
```

여기서 행렬 X와 옵션 Times에 할당된 벡터 T는 주어진 관찰점들을 나타내고, 차원이 $25,000 \times 1$인 벡터 tii는 내삽을 하고자 하는 시점을 나타낸다.

이 MATLAB프로그램을 실행하면, 그림 5.5.12가 그려진다. 이 그림의 상단 그래프는 내삽된 25,000개 표본경로들의 표본평균이다. 이 그래프에서 양끝의 흑색 점들은 시뮬레이트된 값들이고, 가운데 7개 적색 동그라미들은 내삽된 표본경로들의 표본평균들을 나타낸다. 이 그래프에서 알 수 있듯이, 이 표본평균들은 양끝의 시뮬레이트된 값들을 잇는 직선 상에 존재하는 것 같다. 그림 5.5.12의 하단 그래프는 내삽된 표본경로들의 표본분산들을 나타낸다. 내삽을 해도 양끝의 점들은 변함이 없으므로, 이에 해당하는 표본분산은 0이다. 반면에, 이 끝점들에서 멀어져서 가운데로 가까이 갈수록 표본분산은 커진다. ■

예제 5.5.23와 예제 5.5.24에서 사용한 Brown내삽법은 정제(refinement)를 사용하지 않는 방법이다. 정제를 사용하지 않는 내삽법이란 새로이 얻은 정보를 사용하지 않는 방법이다. 반면에, 정제를 사용하는 내삽법(interpolation with refinement)은 내삽에 의해서 얻어진 새로운 관찰점을 기존의 시계열데이터에 추가한 다음 이를 사용해서 다음 내삽을 하는 것이다. 이 경우에 어떤 내삽시점 후의 각 내삽시점에서 발생하는 표본값은 동일하다. 또한, 정제를 사용하는 내삽법의 내삽값들은 정제를 사용하지 않는 내삽법에 비해서 시뮬레이트된 값들을 이은 직선에서 더 떨어져 있으나 표본분산은 0이다.

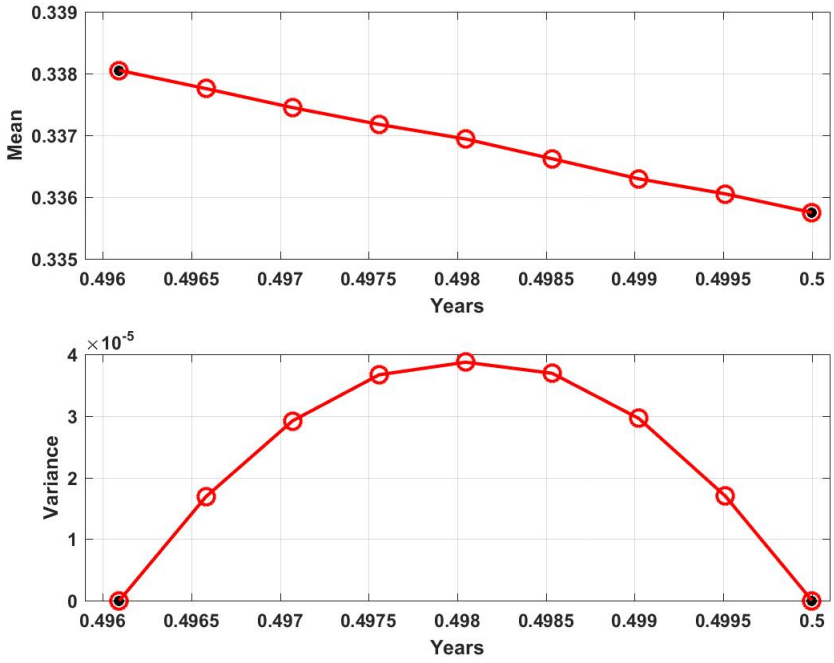


그림 5.5.12. 내삽된 표본경로들의 표본평균과 표본분산

예제 5.5.25 예제 5.5.24에서와 마찬가지로 Brown운동모형에서 표본경로를 한 개 생성하고 이 표본경로에 정제된 내삽법을 적용해서 생성한 25,000개 표본경로들의 표본평균과 표본분산을 구하기 위해서, 다음 MATLAB프로그램 SDEinterpolation103.m을 실행하기로 하자.

```
% ------------------------------------------------------------------------
%  Filename: SDEinterpolation103.m
%  Brownian Interpolation w/o Refinement 2X
% Programmed by CBS
%-------------------------------------------------------------------------
clear all, close all, clc, format rat
% Generating Sample paths of Brownian motion
BM1 = bm(0.3,0.2);
rng(5489,'twister')
dt = 1/256;
[X,T] = BM1.simulate(256,'DeltaTime',dt);
% Interpolating w/ Refinement
midNo = 256/2
nTrials = 25000                % Number of tials at each time
ttMid = (T(midNo):dt/8:T(midNo+1))'
 % Sampling Scheme for the Power-of-Two algorithm
 timess = (1:8)';
 ttMid = NaN(length(timess)+1,1);
 ttMid(1) = T(128)
 ttMid(2) = T(129)
 ndelta = 8;
 jMax = 1;
 iCount = 2+1;
 for kk = 1:3
     ii = ndelta/2;
     for jj = 1:jMax
```

```
        ttMid(iCount) = T(128)+timess(ii)/256/8;
        ii = ii+ndelta;
        iCount = iCount+1;
    end
    jMax = 2*jMax;
    ndelta = ndelta/2;
end
ttMid
plot(1:9,ttMid,'blacko-','linewidth',2,'Markersize',10)
set(gca,'fontsize',11,'fontweigh','bold')
grid on
xlabel('Index'), ylabel('Interpolation Time')
saveas(gcf,'SDEinterpolation103a.jpg')
 % Mean and Variance of Sample paths
format long, figure
XMid = [ X(midNo) X(midNo+1) ]
averInter = zeros(length(ttMid),1);
varInter = zeros(length(ttMid),1);
for ii = 1:length(ttMid)
    tii = ttMid(ii);
    xx = BM1.interpolate(tii(ones(nTrials,1)),X,'Times',T,'Refine',true);
    averInter(ii) = mean(xx(:,1));
    varInter(ii) = var(xx(:,1));
end
subplot(2,1,1)
plot([T(midNo) T(midNo+1)],[X(midNo) X(midNo+1)],'black*-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0.4959 0.5001])
hold on
[SttMid SttMidI] = sort(ttMid)
plot(SttMid,averInter(SttMidI),'redo-','linewidth',2,'Markersize',10)
xlabel('Years'), ylabel('Mean')
grid on
hold off
subplot(2,1,2)
plot([T(midNo) T(midNo+1)],[0 0],'black*','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold')
hold on
plot(SttMid,varInter(SttMidI),'redo-','linewidth',2,'Markersize',10)
xlabel('Years'), ylabel('Variance')
axis([0.4959 0.5001 -10^-25 10^-25 ])
grid on
hold off
saveas(gcf,'SDEinterpolation103b.jpg')
% End of program
% ------------------------------------------------------------------------
```

예제 5.5.24에서와 같이 Brown운동모형 MB1으로부터 표본경로를 1개 생성하였다. 이렇게 생성된 상태벡터 X는 다음 식들을 만족한다.

$$X\left(\frac{127}{256}\right) = 0.338,051, \quad X\left(\frac{128}{256}\right) = 0.335,750 \tag{1}$$

정제된 분할법을 적용하기 위해서 다음과 같은 순서로 시간구간 $[127/256, 128/256]$을 8개 소구간들로 나눈 차원이 9×1인 분할벡터 ttMid를 구하기로 하자. 양끝 시점들을 각각 ttMid(1)과 ttMid(2)로 하고, 이들의 평균을 ttMid(3)으로 한다. 다음으로, ttMid(1)과 ttMid(3)의

평균을 ttMid(4)로 그리고 ttMid(2)와 ttMid(3)의 평균을 ttMid(5)로 한다. 또한, ttMid(1)과 ttMid(4)의 평균을 ttMid(6)로, ttMid(3)과 ttMidd(4)의 평균을 ttMid(7)으로, ttMid(3)와 ttMid(5)의 평균을 ttMid(8)로, 그리고 ttMid(2)와 ttMid(5)의 평균을 ttMid(9)으로 한다. 즉, 분할벡터 ttMid의 원소들을 다음 순서로 구한다.

$$\frac{127}{256} \to \frac{128}{256} \to \frac{255}{512} \to \frac{509}{1024} \to \frac{511}{1024} \to \frac{1017}{2048} \to \frac{1019}{2048} \to \frac{1021}{2048} \to \frac{512}{1025} \quad (2)$$

이렇게 분할벡터 ttMid을 구하는 과정을 그린 것이 그림 5.5.13이다.

정제된 내삽을 하기 위해서는 이 분할시점들의 순서에 따라 내삽해야 한다. 이러한 목표를 달성하기 위해서, 다음 MATLAB명령문을 실행한다.

```
>> xx = BM1.interpolate(tii(ones(nTrials,1)),X,'Times',T,'
Refine',true);
```

여기서 행렬 X와 옵션 Times에 할당된 벡터 T는 주어진 관찰점들을 나타내고, 차원이 $25,000 \times 1$인 벡터 tii는 내삽을 하고자 하는 시점을 나타낸다. 또한, 옵션 Refine에 true를 할당하면, 정제된 내삽을 한다.

이 MATLAB프로그램을 실행하면, 그림 5.5.14이 그려진다. 이 그림의 상단 그래프는 정제된 내삽에 의해서 25,000개 표본경로의 표본평균이다. 이 그래프에서 양끝의 흑색 점들은 시뮬레이트된 값들이고, 가운데 7개 적색 동그라미들은 내삽된 표본경로들의 표본평균들을 나타낸다. 이 그래프에서 알 수 있듯이, 이 표본평균들은 양끝의 시뮬레이트된 값들을 있는 직선에서 떨어져 있다. 정제된 내삽을 하면, 동일한 표본값들이 발생하므로 이에 해당하는 표본분산은 0이다. 하단 그래프에는 내삽된 표본경로들의 표본분산들을 나타낸다. 이 그래프에서 알 수 있듯이, 각 표본분산의 0.5×10^{-25} 보다 작으므로, 각 표본분산은 0임을 확인할 수 있다. ■

5.5.4 이자율모형의 시뮬레이션

이 소절에서는 SDE류의 MATLAB함수를 사용해서 이자율모형으로부터 표본경로를 발생시키기는 과정을 살펴보자.

예제 5.5.26 데이터세트 Global_Idx102_Data의 Euribor 3개월물을 나타내는 시계열데이터 euribor3M의 표본평균과 표본표준편차를 구하고, euribor3M의 시계열산점도를 그

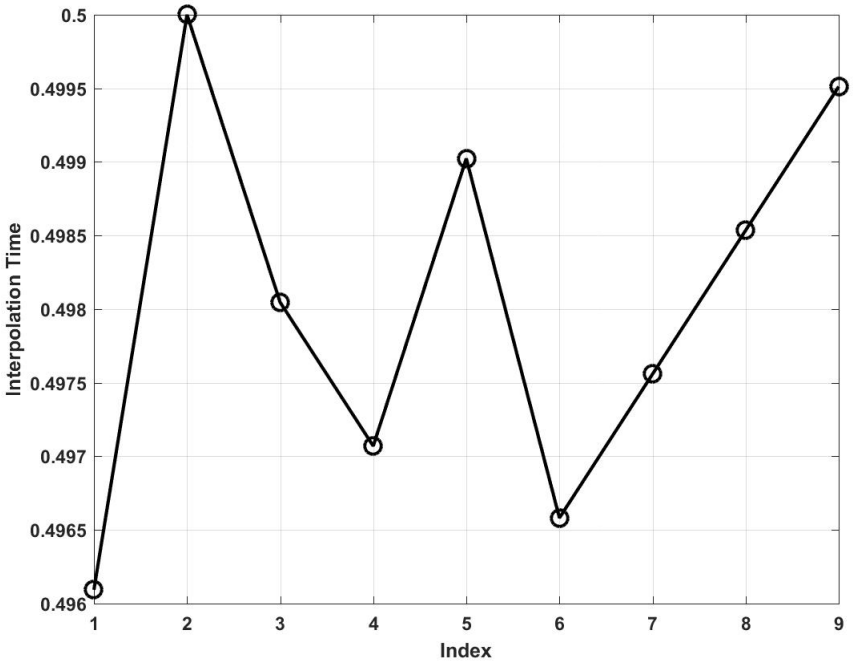


그림 5.5.13. 정제된 내삽을 위한 분할시점들의 순서

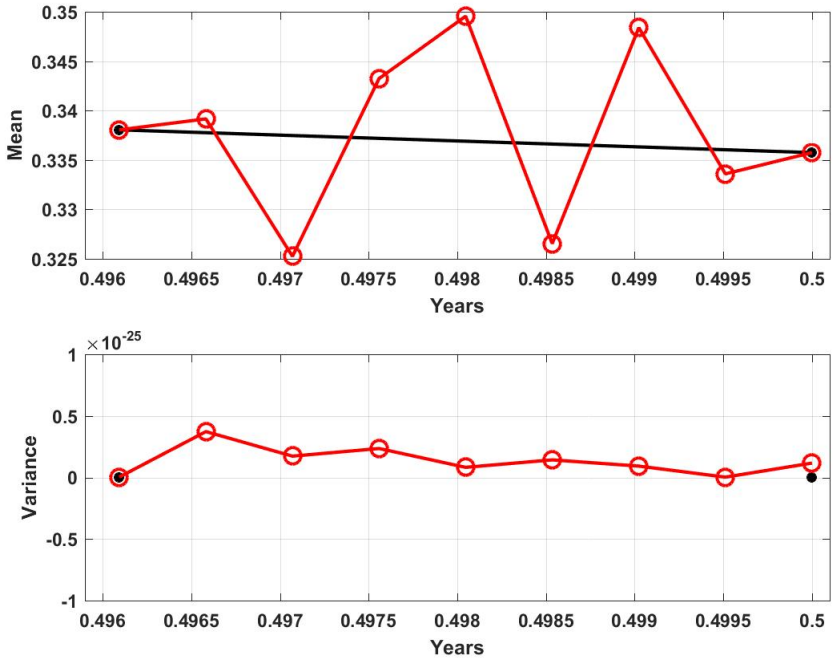


그림 5.5.14. 내삽된 표본경로들의 표본평균과 표본분산

리고, 이 시계열데이터에 적합한 Vasicek모형을 구하기 위해서, 다음 MATLAB프로그램 Data_Global_Euribor101.m을 실행해보자.

```
% ------------------------------------------------------------------------
%  Filename: Data_Global_Euribor101.m
%  Make Global_Euribor3M_Data.mat
%  Based on MATLAB Financial Toolbox Manual
%------------------------------------------------------------------------
clear all, close all, clc, format long
load Global_Idx2_Data
% Descriptive statistics
meanEuribor = mean(euribor3M)
stdEuribor = std(euribor3M)
% Plotting
plot(dates,100*euribor3M,'r-','linewidth',2)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[730889 732791])
DateTick = dates(1:252:end);
DateTickLabel = datestr(DateTick,2);
set(gca,'xtick',DateTick,'xticklabel',DateTickLabel)
xlabel('Date'), ylabel('Daily Yield (%)')
saveas(gcf,'Data_Global_Euribor101.jpg')
% Fitting Vasicek model to the data
regressors = [ ones(length(euribor3M)-1,1) euribor3M(1:end-1) ];
[ coeff CI residuals] = regress(diff(euribor3M),regressors);
coeff
stdres = std(residuals)
dt = 1                                % The time increment is one day.
Vspeed = -coeff(2)/dt                 % Vasicek speed
Vlevel = -coeff(1)/coeff(2)           % Vasicek level
Vsigma = stdres/sqrt(dt)     % Vasicek sigma
% Save the dataset
save Global_Euribor3M_Data101
% End of program
% ------------------------------------------------------------------------
```

이 MATLAB프로그램을 실행하면, 시계열데이터 euribor3M의 표본평균이 7.73×10^{-5} 이고 표본표준편차가 2.25×10^{-5} 임을 알 수 있다. 또한, 이 시계열데이터의 시계열산점도가 그림 5.5.15처럼 그려진다.

Vasicek의 이자율모형은 다음과 같다.

$$dx_t = speed\,[level - x_t]\,dt + \sigma dW_t \tag{1}$$

다음 식들이 성립한다.

$$E\left(x_T \mid x_0\right) = x_0 e^{T \cdot speed} + level\left[1 - e^{T \cdot speed}\right] \tag{2}$$

$$Var\left(x_T \mid x_0\right) = \sigma^2 \frac{1 - e^{T \cdot speed}}{2speed} \tag{3}$$

다음과 같은 선형회귀모형을 살펴보자.

$$y_t = \alpha + \beta x_t + \epsilon_t \tag{4}$$

여기서 ϵ_t는 오차항이고 변수들은 다음과 같다.

$$y_t = dX_t, \quad \alpha = speed \cdot level \cdot dt, \quad \beta = -speed \cdot dt \tag{5}$$

또한, 변동성 σ는 다음 식을 만족한다.

$$\sigma = \sqrt{\frac{Var\left(\epsilon_t\right)}{dt}} \tag{6}$$

따라서, 선형회귀모형 (4)의 추정식과 식 (5)와 식 (6)을 사용해서, 모수들 $speed$, $level$, 그리고 σ의 추정값들을 얻을 수 있다. 우선, 선형회귀모형 (4)를 추정한 결과는 다음과 같다.

$$\hat{\alpha} = 0.000,000,137, \quad \hat{\beta} = -0.002,288,541 \tag{7}$$

또한, 오차항의 표본표준편차는 4.776×10^{-7}이다. 따라서, 다음 식들이 성립한다.

$$\hat{speed} = 0.002,288,541, \ \hat{level} = 6.004,237 \times 10^{-5}, \hat{\sigma} = 4.776,367 \times 10^{-7} \tag{8}$$

기존의 데이터와 더불어 새로이 생성된 데이터들이 MATLAB데이터파일 Global_Euribor3M_Data101.mat에 저장된다. ■

MATLAB함수 hwv.m을 사용해서, Hull-White-Vasick이자율모형으로부터 표본경로를 발생시켜보자.

예제 5.5.27 Hull-White-Vasicek이자율모형으로부터 표본경로를 생성하기 위해서, 다음 MATLAB프로그램 SDE_Interest101.m을 실행해보자.

```
% ------------------------------------------------------------------------
%  Filename: SDE_Interest101.m
%  Simulating Interest Rates 1
%-------------------------------------------------------------------------
clear all, close all, clc, format long
load Global_Euribor3M_Data101
% Simulating the fitted model
HWV1 = hwv(Vspeed, Vlevel,Vsigma,'StartState',euribor3M(end))
% Sampling scheme for the Power-of-Two algorithm
Tend = 8
TT = (1:Tend)';
ttMid = zeros(length(TT)+1,1);
ttMid(1) = HWV1.StartTime;
ttMid(2) = Tend;
delta = Tend;
```

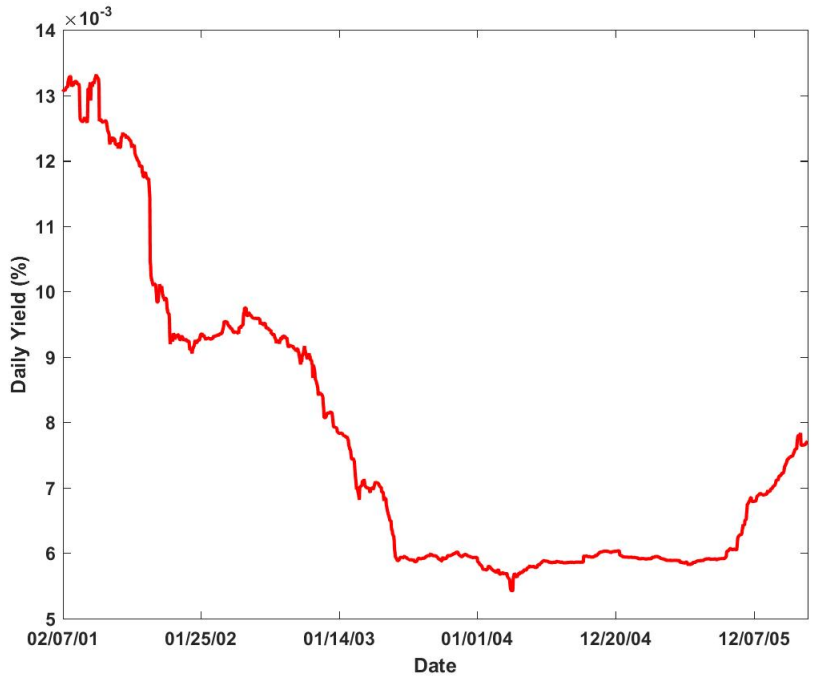


그림 5.5.15. Euribor3M의 시계열산점도

```
jMax = 1;
iCount = 3;
for kk = 1:log2(Tend)
    ii = delta/2;
    for jj = 1:jMax
        ttMid(iCount) = TT(ii);
        ii = ii+delta;
        iCount = iCount+1;
    end
    jMax = 2*jMax;
    delta = delta/2;
end
disp('ttMid = '), disp(ttMid')
% initializing the time series grid
Xstart = HWV1.StartState
Xend = Xstart*exp(-Vspeed*Tend)+Vlevel*(1-exp(-Vspeed*Tend))
X = [ Xstart; Xend ];
% Generate three sample paths
nTrials = 3
rng(5489,'twister')
Xinitial = X(:,:,ones(nTrials,1))
Tinitial = [HWV1.StartTime Tend]
Y = HWV1.interpolate(ttMid,Xinitial,'Times',Tinitial,'Refine',true);
% Plotting the Sample paths
figure
[ttMid,itt] = sort(ttMid);
Y = squeeze(Y);
disp('[ index Y ] =  '), disp([ (1:Tend+1)' Y ])
Ys = Y(itt,:);
disp('[ itt Ys ] = '), disp([ itt Ys ])
plot(ttMid,100*Ys,'-o','linewidth',2,'MarkerSize',10)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 8])
hold on
```

```
plot(ttMid([1 end]),100*Ys([1 end],1),'black.','MarkerSize',30)
xlabel('Interpolation Time')
ylabel('Yield (%)')
saveas(gcf,'SDE_Interest101.jpg')
% End of program
% ------------------------------------------------------------------------
```

첫 번째 단계로 HWV모형 HWV1을 생성하기 위해서, 다음 MATLAB명령문을 실행한다.

```
>> HWV1 = hwv(Vspeed,Vlevel,Vsigma,'StartState',euribor3M(end))
```

여기서 입력모수들 Vspeed, Vlevel 그리고 Vsigma는 각각 예제 5.5.26에서 추정한 Vasicek 모형의 모수추정값들 $\hat{speed}, \hat{level}$ 그리고 $\hat{\sigma}$이다. 또한, 옵션 StartState에는 시계열데이터 euribor3M의 마지막 관찰값을 할당하였다. 이 MATLAB명령문을 실행한 결과는 다음과 같다.

```
HWV1 =
   Class HWV: Hull-White/Vasicek
   ----------------------------------------
     Dimensions: State = 1, Brownian = 1
   ----------------------------------------
      StartTime: 0
     StartState: 7.70408e-05
    Correlation: 1
          Drift: drift rate function F(t,X(t))
      Diffusion: diffusion rate function G(t,X(t))
     Simulation: simulation method/function simByEuler
          Sigma: 4.77637e-07
          Level: 6.00424e-05
          Speed: 0.00228854
```

주가과정의 표본경로를 생성할 때는 시작 시점의 고정된 상태에서 미래시점으로 향하면서 시뮬레이션을 하였다. 따라서, Brown운동으로부터 Gauss주가과정을 생성할 수 있다. 그러나, 이자율과정은 다른 확률과정을 할인하는 할인과정(discounted process)으로 많이 사용하므로 시작 시점의 상태 뿐 아니라 마지막 시점의 상태도 고정시켜야 하는 경우가 많다. 따라서, Gauss이자율과정의 표본경로를 생성할 때는 Brown운동보다는 Brown다리가 더 적절하다. 즉, Brown내삽을 적용하기로 하자.

Brown다리를 사용해서 표본경로를 발생하기 위해서는 먼저 양끝 시점들을 지정하고 이로부터 내삽할 시점들을 정해야 한다. 이 예제에서는 정제된 Brown내삽법을 적용하기로 하자. 우선 양끝 시점들을 각각 ttMid(1) = HMV1.StartTime과 ttMid(2) = 8로 한다. 여기서 HMV1.StartTime는 디폴트인 0이다. 다음으로 ttMid(1)과 ttMid(2)의 평균 4를 ttMid(3)로, ttMid(1)과 ttMid(3)의 평균 2를 ttMid(4)로, ttMid(2)와 ttMid(3)의 평균 6을

ttMid(5)로, ttMid(1)과 ttMid(4)의 평균 1을 ttMid(6)로, ttMid(3)과 ttMidd(4)의 평균 3을 ttMid(7)으로, ttMid(3)와 ttMid(5)의 평균 5를 ttMid(8)로, 그리고 ttMid(2)와 ttMid(5)의 평균 7을 ttMid(9)으로 한다. 즉, 분할벡터 ttMid의 원소들을 다음 순서로 구한다.

$$0 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 6 \rightarrow 1 \rightarrow 3 \rightarrow 5 \rightarrow 7 \tag{1}$$

초기시점인 ttMid(1)에서 상태값 Xstart로는 HMV1.StartState인 $7.704,084 \times 10^{-5}$를 할당하고, 마지막 시점인 ttMid(8)에서 상태값 Xend로는 euribor3M의 Vasicek기대값을 사용한다. 이 Xend를 구하기 위해서 다음 MATLAB명령문을 실행한다.

```
>> Xend = Xstart*exp(-Vspeed*Tend)+Vlevel*(1-exp(-Vspeed*Tend));
```

이 Xend는 $7.673,246 \times 10^{-5}$이다.

정제된 내삽을 하기 위해서는 먼저 MATLAB명령문 interpolate.m에 대입한 입력모수들을 살펴보자. 주어진 양끝 시점들의 벡터가 Tinitial이고 이 시점들에서 상태값들의 벡터가 Xinitial이다. 이 Xinitial을 구하기 위해서 다음 MATLAB명령문을 실행하자.

```
>> Xinitial = X(:,:,ones(nTrials,1))
```

여기서 X(:,:,ones(nTrials,1))은 차원이 $2\times2\times3$인 행렬로서 각 k에 대해서 행렬 Xinitial(:,:,k)는 다음과 같다.

```
1.0e-04 *
0.770408372459119
0.767324553418138
```

정제된 내삽을 하기 위해서는 식 (2)의 순서에 따라 내삽을 해야 한다. 이러한 목표를 달성하기 위해서, 다음 MATLAB명령문을 실행한다.

```
>> Y = HWV1.interpolate(ttMid,Xinitial,'Times',Tinitial,'Refine',
  true);
```

여기서 입력모수 ttMid는 내삽할 시점들을 나타내는 벡터이다. 또한, 옵션 Refine에 true를 할당했으므로, 이 MALAB명령문은 정제된 내삽을 하도록 한다.

이 MATLAB프로그램을 실행하면, 그림 5.5.16이 그려진다. 그림 5.5.16에서 알 수 있듯이, 시작 시점에서 상태값들이 동일하고 마지막 시점에서 상태값들도 동일한 표본경로들이 3개 그려진다. ■

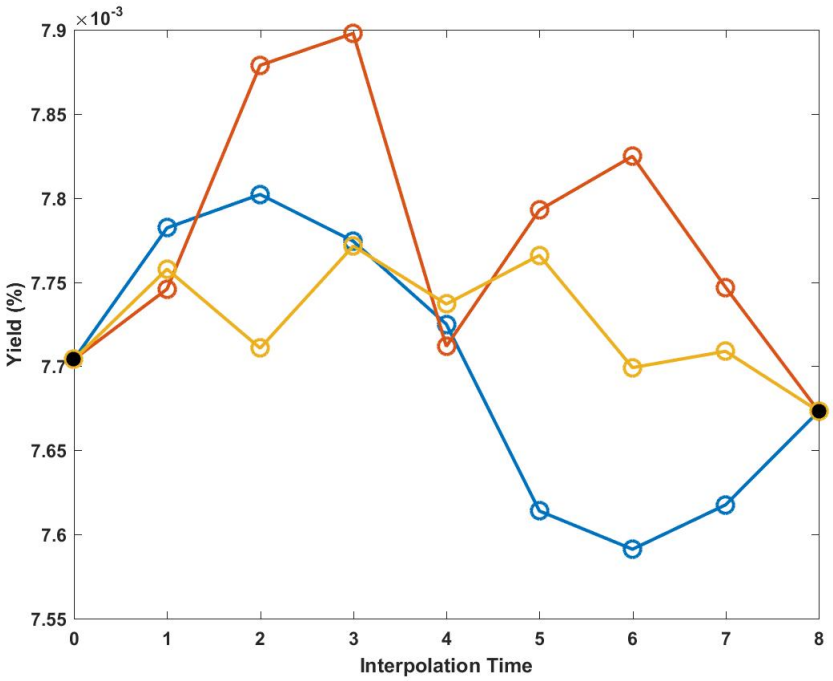


그림 5.5.16. 정제된 내삽을 사용해서 구한 표본경로

이자율이나 금융자산가치는 양(positive)이어야 한다. MATLAB의 SDE류 함수에서 GBM의 symBySolution을 제외하고는 생성된 표본경로가 항상 양이라는 보장이 없다. 따라서, 독자 스스로 표본경로가 항상 양이 되도록 프로그램을 해야한다. MATLAB을 사용해서 시뮬레이션이나 내삽을 하는 경우, 이러한 목적을 달성하기 위해서는 SDE류 함수의 옵션 Processes에 적절한 함수들 또는 배경과정들(a sequence of functions or background processes)을 할당한다. 이러한 함수 또는 배경과정을 과정함수(processing function)라 부른다. MATLAB은 각 표본시간의 소구간의 끝점에서(at the end of sample time period) 과정함수들을 반영해서 난수를 발생하거나 내삽을 한다. 이러한 과정함수는 경계조건을 명시하거나, 통계량을 축적하거나, 그림을 그리거나, 또는 경로의존형 옵션의 가치를 평가하는데 유용한 도구이다. 다음 예제는 과정함수를 사용해서 양인 표본경로를 생성하는 것이다.

예제 5.5.28 다음과 같은 CIR모형(Cox-Ingersoll-Ross object)(Cox-Ingersoll-Ross object)을 살펴보자.

$$dx_t = S\,[L - x_t]\,dt + \sigma x_t^{1/2} dW_t \tag{1}$$

여기서 모수들은 다음 Feller조건을 만족한다고 하자.

$$2SL \geq \sigma^2 \tag{2}$$

이 Feller조건이 만족되면, 다음 식이 성립함이 알려져 있다.

$$x_t > 0 \ \ a.s. \ \ (t > 0) \tag{3}$$

확률미분방정식 (1)에서 식 (3)을 만족하는 표본경로를 생성하기 위해서, 다음 MATLAB 프로그램 SDE_Interest102.m을 실행해보자.

```
% ------------------------------------------------------------------------
%  Filename: SDE_Interest102.m
%  Simulating Interest Rates 2
%  Based on Example of MATLAB Financial Toolbox
%------------------------------------------------------------------------
clear all, close all, clc, format long
% Simulating daily short rates of the CIR  model
rng(14151617,'twister')
CIR1 = cir(0.25, @(t,X) 0.1, 0.2,'StartState',0.02)
[X,T] = CIR1.simByEuler(250,'DeltaTime',1/250,'nTrials',5);
disp(' ')
disp('[T(196:201)  X(196:201,1,4)] = ')
disp([T(196:201)  X(196:201,1,4)])
% Repeating the simulation w/ a Processing function
rng(14151617,'twister')
[Y,T] = CIR1.simByEuler(250,'DeltaTime',1/250,'nTrials',5, ...
                        'Processes', @(t,X) abs(X));
disp(' ')
disp('[T(196:201)  Y(196:201,1,4)] = ')
disp([T(196:201)  Y(196:201,1,4)])
whos
% Plotting the Sample paths
subplot(2,1,1)
plot(T,100*abs(X(:,1,4)),'green-',T,100*Y(:,1,4),'black:','linewidth',2.5)
set(gca,'fontsize',11,'fontweigh','bold')
legend('Negative/Complex Rates','Positive Rates','Location','Best')
xlabel('Years'), ylabel('Short Rate (%)')
axis([ 150/250 250/250 0 0.4 ])
subplot(2,1,2)
plot(T,100*abs(X(:,1,4)),'green-',T,100*Y(:,1,4),'black:','linewidth',2.5)
set(gca,'fontsize',11,'fontweigh','bold')
legend('Negative/Complex Rates','Positive Rates',1)
xlabel('Years'), ylabel('Short Rate (%)')
axis([ 195/250 215/250 0 0.15 ])
saveas(gcf,'SDE_Interest102.jpg')
% End of program
% ------------------------------------------------------------------------
```

첫 번째 단계로 CIR모형 CIR1을 생성하기 위해서, 다음 MATLAB명령문을 실행한다.

```
>> CIR1 = cir(0.25,@(t,X) 0.1, 0.2, "StartState, 0.02);
```

이 명령문은 다음 확률미분방정식을 모형화하는 것이다.

$$dx_t = 0.25\,[0.1 - x_t]\,dt + 0.2x_t^{1/2}dW_t \tag{4}$$

식 (4)의 모수들은 다음과 같이 Feller조건을 만족한다.

$$2 \cdot 0.25 \cdot 0.1 = 0.05 \geq 0.04 = 0.2^2 \tag{5}$$

이 MATLAB명령문을 실행한 결과는 다음과 같다.

```
CIR1 =
   Class CIR: Cox-Ingersoll-Ross
   ----------------------------------------
     Dimensions: State = 1, Brownian = 1
   ----------------------------------------
      StartTime: 0
     StartState: 0.02
    Correlation: 1
          Drift: drift rate function F(t,X(t))
      Diffusion: diffusion rate function G(t,X(t))
     Simulation: simulation method/function simByEuler
          Sigma: 0.2
          Level: function @(t,X)0.1
          Speed: 0.25
```

표본경로를 생성하기 위해서, 다음 MATLAB명령문들을 실행한다.

```
>> rng(14151617,'twister')
   >> [X,T] = CIR1.simByEuler(250,'DeltaTime',1/250,'nTrials',5);
```

이 MATLAB명령문에서는 시뮬레이션방법으로 simBYEuler를 할당하였고, 첫 번째 모수로 250을, 옵션 DeltaTime에 1/250을, 그리고 옵션 nTrials에는 5를 할당하였다. 즉, 이 MATLAB명령문은 1년의 거래일(trading days)을 250일로 보아 소구간의 길이를 1/250으로 하고, 초기시점을 포함해서 251개 관찰시점들로 구성된 표본경로들을 5개 생성하라는 것이다. 이 MATLAB명령문이 실행되면, 출력변수 X는 차원이 $251 \times 1 \times 5$인 행렬이고 출력변수 T는 차원이 251×1인 행렬이다. 그 중에서 제4번째 표본경로의 시간구간 [0.780.80]에서 이들을 출력한 결과는 다음과 같다.

```
[T(196:201)  X(196:201,1,4)] =
  0.780000000000001 + 0.000000000000000i  0.000356658520226 +
    0.000000000000000i
  0.784000000000001 + 0.000000000000000i  0.000242196341945 +
    0.000000000000000i
  0.788000000000001 + 0.000000000000000i -0.000027082713566 +
    0.000000000000000i
  0.792000000000001 + 0.000000000000000i  0.000072944369147 +
    0.000047133850793i
  0.796000000000001 + 0.000000000000000i  0.000179713931282 +
    0.000049105053470i
  0.800000000000001 + 0.000000000000000i  0.000239894193698 +
    0.000043737803033i
```

이 출력물에서 알 수 있듯이, 상태값이 음수뿐 아니라 복소수가 되기도 한다. 이는 CIR모형에 $\sqrt{x_t}$ 항이 포함되어 발생하는 것이다.

과정함수를 사용해서 양인 표본경로를 생성하기 위해서, 다음 MATLAB명령문을 실행한다.

```
>> rng(14151617,'twister')
>> [Y,T] = CIR1.simByEuler(250,'DeltaTime',1/250,'nTrials',5, ...
                           'Processes',@(t,X) abs(X));
```

이 MATLAB명령문에서는 옵션 Processes에 과정함수 @(t,X) abs(X)를 지정해서 생성되는 표본값이 음수가 되는 것을 방지했다. 여기서 유의할 점은 과정함수를 독립변수들 t와 X의 함수로 지정할 수 있으나, 이자율 X의 상태벡터만이 사용된다는 것이다. 제4번째 표본경로의 시간구간 [0.780.80]에서 이들을 출력한 결과는 다음과 같다.

```
[T(196:201)  Y(196:201,1,4)] =
   0.780000000000001   0.000356658520226
   0.784000000000001   0.000242196341945
   0.788000000000001   0.000027082713566
   0.792000000000001   0.000174189481646
   0.796000000000001   0.000284118607299
   0.800000000000001   0.000334443377430
```

이 출력물에서 알 수 있듯이, 과정함수를 사용해서 생성한 표본경로는 항상 양이다.

이 MATLAB프로그램을 실행하면, 그림 5.5.17이 그려진다. 그림 5.5.17의 각 그래프에서 녹색 실선은 과정함수를 사용하지 않고 발생시킨 표본경로이고, 흑색 점선은 과정함수를 사용해서 발생시킨 표본경로이다. 그림 5.5.17의 상단 그래프는 시간구간 [0.61]에서 표본경로들을 그린 것이고, 하단 그래프는 시간구간 [0.750.86]에서 표본경로들을 그린 것이다. ■

5.5.5 층화추출

제5.5.3소절에서 다룬 Brown내삽을 확장한 층화추출법(stratified sampling method)(stratified sampling method)을 적용해서 확률미분방정식으로부터 표본경로들을 생성할 수 있다. 층화추출법을 두 단계로 나누어 실행한다. 첫째, 마지막 시점을 표본경로들의 개수에 해당하는 층들(strata)로 나눈다. 즉, 주어진 초기시점의 상태값에서 출발한 표본경로가 마지막 시점에서 취할 수 있는 상태값들을 먼저 구하고, 각 상태값을 하나의 층으로 간주한다. 즉, 지지대가 $[0,1]$인 정규분포에서 표본경로들 개수만큼 난수들을 생성한다. 이 난수들에 역함수법을

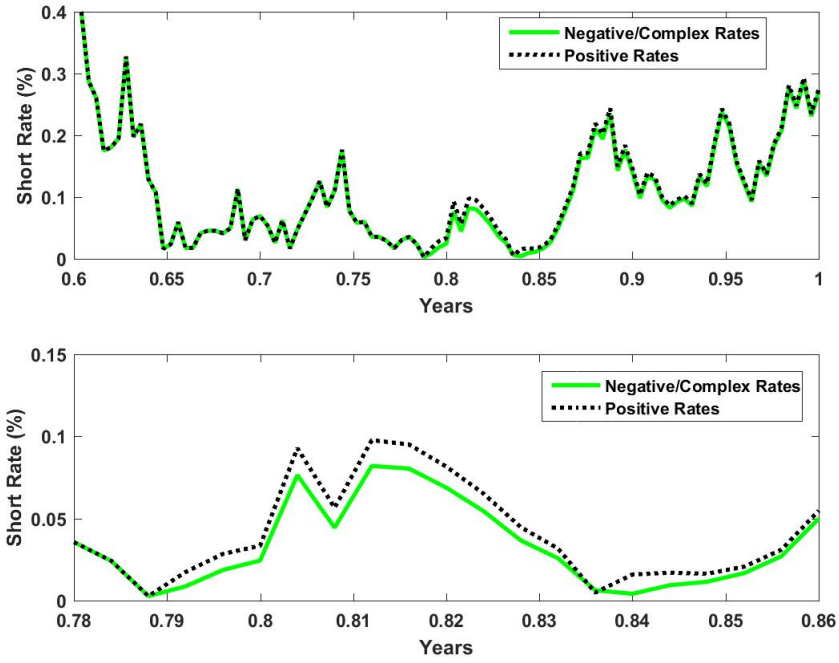


그림 5.5.17. 이자율의 표본경로들

적용해서 표본경로의 마지막 시점에서 상태값들을 구한다. 이를 비례표본추출(proportional sampling)(proportional sampling)이라 부른다. 둘째, 초기시점의 상태값과 마지막 시점의 상태값을 연결하는 Brown내삽을 해서 표본경로를 생성한다. 층화추출법은 분산감소법의 일종이므로, 이러한 방법을 적용해서 표본경로들의 분산을 감소시킬 수 있다.

예제 5.5.29 다음과 같은 Brown운동에 표본경로를 생성하기로 하자.

$$dx_t = dW_t \tag{1}$$

층화추출법을 적용해서 확률미분방정식 (1)에서 표본경로를 생성하기 위해서, 다음 MATLAB 프로그램 SDE_StratifiedSampling101.m을 실행해보자.

```
% ------------------------------------------------------------------------
%  Filename: SDE_StratifiedSampling101.m
%  Stratified Sampling of Brownian Motion 1
%-------------------------------------------------------------------------
clear all, close all, clc, format short
% Simulating the standard Brownian paths by the stratified sampling ftn
dt = 1/252                    % 1 year = 21*12 trading days
nPeriods = 2*21               % Simulating  for 2 months
tau = nPeriods*dt             % Tenor in years
nPaths = 5                    % Number of simulated paths
BM1 = bm(0,1,'StartState',0)
sampleTimes = cumsum([BM1.StartTime; dt(ones(nPeriods,1))]);
```

```
[sampleTimes(1) sampleTimes(end)]
whos sampleTimes
rng(5489,'twister')
z = Example_StratifiedRNG(nPaths,sampleTimes)
X = BM1.simulate(nPeriods,'DeltaTime',dt,'nTrials',nPaths,'Z',z);
whos X
% Reordering 3D output to a 2-D equivalent array
X = squeeze(X);
whos X
disp(' '), disp('X(end,:) = ')
disp(X(end,:))
% Verifying the stratification
rng(5489,'twister')
U = ((1:nPaths)'-1+rand(nPaths,1))'/nPaths
BMtau = norminv(U)*sqrt(tau)    % Stratified Brownian Motion
% Plotting
plot(sampleTimes,X,'linewidth',2.5)
set(gca,'fontsize',11,'fontweigh','bold','xlim',[0 tau])
xlabel('Years'), ylabel('Brownian State')
hold('on')
plot(tau,BMtau,'black.','Markersize',20)
hold('off')
saveas(gcf,'SDE_StratifiedSampling101.jpg')
% End of program
% ------------------------------------------------------------------------
```

첫 번째 단계로 BM모형 BM1을 생성하기 위해서, 다음 MATLAB명령문을 실행한다.

```
>> BM1 = bm(0, 1, 'StartState', 0);
```

이 명령문은 확률미분방정식 (1)을 모형화하는 것이다. 이 MATLAB명령문을 실행한 결과는 다음과 같다.

```
BM1 =
   Class BM: Brownian Motion
   ----------------------------------------
     Dimensions: State = 1, Brownian = 1
   ----------------------------------------
      StartTime: 0
     StartState: 0
    Correlation: 1
          Drift: drift rate function F(t,X(t))
      Diffusion: diffusion rate function G(t,X(t))
     Simulation: simulation method/function simByEuler
             Mu: 0
          Sigma: 1
```

두 번째 단계로 층화를 사용해서 표본경로를 생성한다. 우선 StartTime인 0에서 출발해서 dt씩 tau까지 증가하는 원소들을 포함하는 열벡터 sampleTimes를 만든다. 이 열벡터의 길이는 nPeriods+1이다. MATLAB함수 Example_StratifiedRNG.m은 층화를 적용해서 마지막 시점에서 상태값을 시뮬레이트하기 위한 것이다. 이 MATLAB함수의 입력변수들로

표본경로들의 개수와 상태값들을 시뮬레이트하는 시점들의 벡터이다. 표본경로를 생성하기 위해서, 다음 MATLAB명령문들을 실행한다.

```
>> X = BM1.simulate(nPeriods,'DeltaTime',dt,'nTrials',nPaths,'Z',z);
>> X = squeeze(X);
```

여기서 첫 번째 MATLAB명령문에 의해 출력되는 변수 X는 차원이 $43 \times 1 \times 5$인 행렬이다. 또한, MATLAB함수 squeeze.m을 사용해서, 출력변수 X의 차원을 43×5 으로 압축한다. 마지막 시점에서 행렬 X의 관찰값들을 출력하면 다음과 같다.

```
X(end,:) =
   -0.4011    -0.1235    -0.0768     0.3189     0.5920
```

세 번째 단계로 층화과정을 확인하기로 하자. 우선, 다음과 같은 층화된 일양난수들의 열벡터를 생성한다.

```
>> U = ((1:nPaths)'-1+rand(nPaths,1))/nPaths
```

이렇게 생성된 열벡터의 길이는 nPath이고 제j번째 원소는 구간 $\left(\frac{j-1}{\texttt{nPaths}}, \frac{j}{\texttt{nPaths}}\right)$에 속한다. 이 MATLAB명령문을 실행한 결과는 다음과 같다.

```
U =
    0.1629     0.3812     0.4254     0.7827     0.9265
```

네 번째 단계로 이 U에 역변환법을 적용해서 표본경로의 마지막 시점에서 상태값을 구하기 위해서, 다음 MATLAB명령문을 실행하자.

```
>> BMtau = norminv(U)*sqrt(tau)
```

이 MATLAB명령문을 실행한 결과는 다음과 같다.

```
BMtau =
   -0.4011    -0.1235    -0.0768     0.3189     0.5920
```

앞에서 구한 벡터 X(end,:)와 벡터 BMtau가 동일함을 알 수 있다.

이 MATLAB프로그램을 실행하면, 그림 5.5.18이 그려진다. 그림 5.5.18에서 시점 0.1667의 흑색 점들은 벡터 BMtau의 원소들을 나타내고 표본경로들은 행렬 X를 나타낸다. 또한, 벡터 BMtau를 발생시키는 과정과 표본경로들을 나타내는 행렬 X를 생성하는 과정에서

동일한 알고리즘을 사용하므로, 행렬 X가 앞에서 기술한 층화추출법에 의해서 생성되었음을 확인할 수 있다. ■

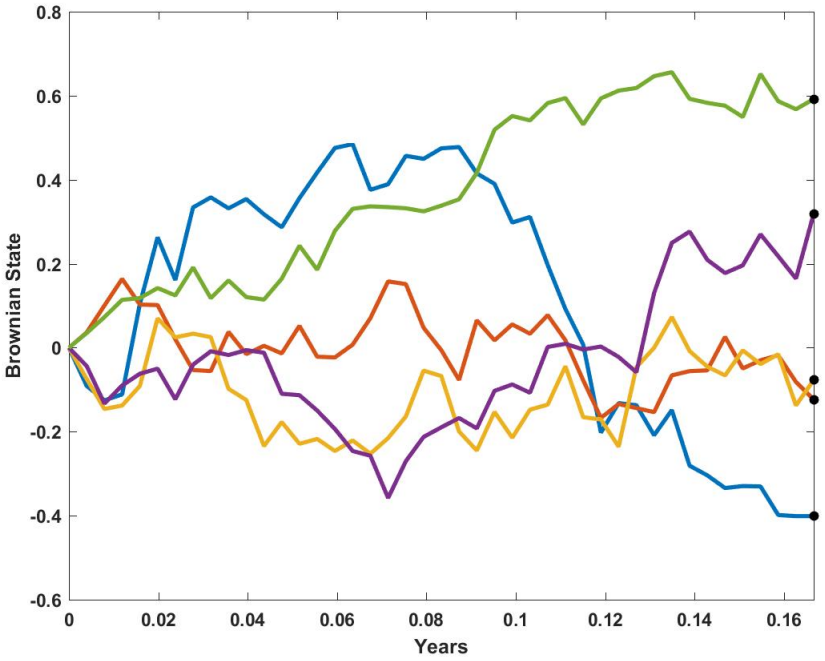


그림 5.5.18. 층화추출법에 의해서 생성된 Brown운동의 표본경로

이러한 층화추출법을 추세계수와 확산계수가 상수들인 기하Brown운동에도 적용할 수 있다.

예제 5.5.30 캐나다 대형주지수인 TSX가 다음과 같은 Brown운동모형을 따른다고 하자.

$$dx_t = rx_t + \sigma x_t dW_t \tag{1}$$

여기서 상수 r은 Euribor 3개월물의 평균수익률이라 하자. 층화추출법을 적용해서 확률미분방정식 (1)에서 표본경로를 생성하기 위해서, 다음 MATLAB프로그램 SDE_StratifiedSampling102.m을 실행해보자.

```
% --------------------------------------------------------------------
%  Filename: SDE_StratifiedSampling102.m
%  Stratified Sampling 2
%--------------------------------------------------------------------
clear all, close all, clc, format short
% Making the TSX return process and constant interest rate
load Global_Euribor3M_Data101
returns1 = returns(:,5);                    % Return of TSX (Canadian index)
```

```
whos returns1
sigma = std(returns1)*sqrt(252)
EuriRate = mean(360*log(1+euribor3M))     % Average of Euribor 3M
% Simulating geometric Brownian motion
GBM1 = gbm(EuriRate,sigma,'StartState',100)
nPaths = 10
dt = 1/252
nPeriods = 21*3
% nSteps = 1
nSteps = 1
sampleTimes1 = cumsum([GBM1.StartTime;dt(ones(nPeriods*nSteps,1))/nSteps]);
whos sampleTimes1
disp(' ')
disp('sampleTimes1(1) sampleTimes1(2) sampleTimes1(end) = ')
disp([sampleTimes1(1) sampleTimes1(2) sampleTimes1(end)])
rng(5489,'twister')
z = Example_StratifiedRNG(nPaths,sampleTimes1)
[Y1 Times1] = GBM1.simBySolution(nPeriods,'nTrials',nPaths, ...
                                 'DeltaTime',dt,'nSteps',nSteps,'Z',z);
Y1 = squeeze(Y1);
whos Times1 Y1
% nSteps = 4
nSteps = 4
sampleTimes4 = cumsum([GBM1.StartTime;dt(ones(nPeriods*nSteps,1))/nSteps]);
whos sampleTimes4
disp(' ')
disp('sampleTimes4(1) sampleTimes4(2) sampleTimes4(end) = ')
disp([sampleTimes4(1) sampleTimes4(2) sampleTimes4(end)])
rng(5489,'twister')
z = Example_StratifiedRNG(nPaths,sampleTimes4);
[Y4 Times4] = GBM1.simBySolution(nPeriods,'nTrials',nPaths, ...
                                 'DeltaTime',dt,'nSteps',nSteps,'Z',z);
Y4 = squeeze(Y4);
whos Times4 Y4
% Plotting
subplot(1,2,1)
plot(Times1,Y1,'linewidth',2.5)
set(gca,'fontsize',11,'fontweigh','bold')
axis([0 0.25 80 135])
xlabel('Years'), ylabel('Index with nSteps = 1')
subplot(1,2,2)
plot(Times1,Y4,'linewidth',2.5)
set(gca,'fontsize',11,'fontweigh','bold')
axis([0 0.25 80 135])
xlabel('Years'), ylabel('Index with nSteps = 4')
saveas(gcf,'SDE_StratifiedSampling102.jpg')
% End of program
% --------------------------------------------------------------------------
```

첫 번째 단계로, MATLAB데이터세트 Global_Euribor3M_Data101.mat를 불러들여서 TSX지수의 로그수익률을 벡터 returns1에, 이 로그수익률의 일별변동성(daily volatility)을 스칼라 sigma에, 그리고 Euribor 3개월물의 일별평균수익률 스칼라 EuriRate에 저장한다. 그 결과는 다음과 같다.

```
      Name             Size                Bytes  Class     Attributes
  returns1          1358x1                10864  double
```

```
sigma =
    0.1812

EuriRate =
    0.0278
```

두 번째 단계로, GBM모형 GBM1을 생성하기 위해서, 다음 MATLAB명령문을 실행한다.

```
>> GBM1 = gbm(EuriRate,sigma,'StartState',100)
```

이 명령문은 확률미분방정식 (1)을 모형화하는 것이다. 이 MATLAB명령문을 실행한 결과는 다음과 같다.

```
GBM1 =
   Class GBM: Generalized Geometric Brownian Motion
   -------------------------------------------------
     Dimensions: State = 1, Brownian = 1
   -------------------------------------------------
      StartTime: 0
     StartState: 100
    Correlation: 1
          Drift: drift rate function F(t,X(t))
      Diffusion: diffusion rate function G(t,X(t))
     Simulation: simulation method/function simByEuler
         Return: 0.0278117
          Sigma: 0.181217
```

세 번째 단계로, 확률방정식 (1)로부터 표본경로를 발생시키기 위해서, 먼저 표본경로들의 개수 nPath가 10, 시뮬레이션하는 시점들의 간격 dt가 1/252, 그리고 시뮬레이션하는 시점들의 개수 nPeriods가 63으로 할당하자. 다음으로 이웃하는 시뮬레이션하는 시점들의 소구간을 균등하게 nSteps으로 나누기로 하자. 이렇게 나누어진 소구간 내의 nSteps-1개 중간시점들에서도 상태값들을 시뮬레이트함으로써 좀 더 정제된(refined) 표본경로를 얻을 수 있다. 여기서는 nSteps = 1이므로, 실제로 각 소구간이 더 작은 소구간들로 나누어지지 않는다. 이렇게 구성된 표본값을 생성하는 시점들의 벡터를 생성하기 위해서, 다음 MATLAB 명령문을 실행한다.

```
>> sampleTimes1 = cumsum([GBM1.StartTime; ...
                          dt(ones(nPeriods*nSteps,1))/nSteps]);
```

이 MATLAB명령문을 실행해서 생성한 길이가 64인 열벡터 sampleTimes1의 첫 번째, 두 번째 그리고 마지막 원소들은 각각 0, 0.0040 그리고 0.2500이다. 다음 MATLAB함수 Example_StratifiedRNG.m은 층화를 적용해서 마지막 시점에서 상태값을 시뮬레이트하기 위한 것이다. 표본경로를 생성하기 위해서, 다음 MATLAB명령문들을 실행한다.

```
>> [Y1 Times1] = GBM1.simBySolution(nPeriods,'nTrials',nPaths, ...
                                'DeltaTime',dt,'nSteps',nSteps,'Z',z);
>> Y1 = squeeze(Y1);
```

이렇게 생성된 행렬들 Times1과 Y1의 속성은 다음과 같다.

```
Name          Size              Bytes  Class     Attributes
 Times1       64x1                512  double
 Y1           64x10              5120  double
```

네 번째 단계로, 확률방정식 (1)로부터 nSteps이 4인 표본경로를 발생시키기로 하자. 따라서, 각 소구간은 4개의 더 작은 소구간들로 나누어진다. 이렇게 구성된 표본값을 생성하는 시점들의 벡터를 생성하기 위해서, 다음 MATLAB명령문을 실행한다.

```
>> sampleTimes4 =  ...
       cumsum([GBM1.StartTime;dt(ones(nPeriods*nSteps,1))/nSteps]);
```

이 MATLAB명령문을 실행해서 생성한 길이가 253인 열벡터 sampleTimes4의 첫 번째, 두 번째 그리고 마지막 원소들은 각각 0, 0.0010 그리고 0.2500이다. 다음 MATLAB함수 Example_StratifiedRNG.m은 층화를 적용해서 마지막 시점에서 상태값을 시뮬레이트하기 위한 것이다. 표본경로를 생성하기 위해서, 다음 MATLAB명령문들을 실행한다.

```
>> [Y4 Times4] = GBM1.simBySolution(nPeriods,'nTrials',nPaths, ...
                                'DeltaTime',dt,'nSteps',nSteps,'Z',z);
>> Y4 = squeeze(Y4);
```

이렇게 생성된 행렬들 Times4와 Y4의 속성은 다음과 같다.

```
Name          Size              Bytes  Class     Attributes
Times4        64x1                512  double
Y4            64x10              5120  double
```

이 MATLAB프로그램을 실행하면, 그림 5.5.19이 그려진다. 그림 5.5.19의 좌측 그래프에는 모수 nSteps가 1인 표본경로들이, 그리고 우측 그래프에는 모수 nSteps가 4인 표본경로들이 그려져 있다. ■

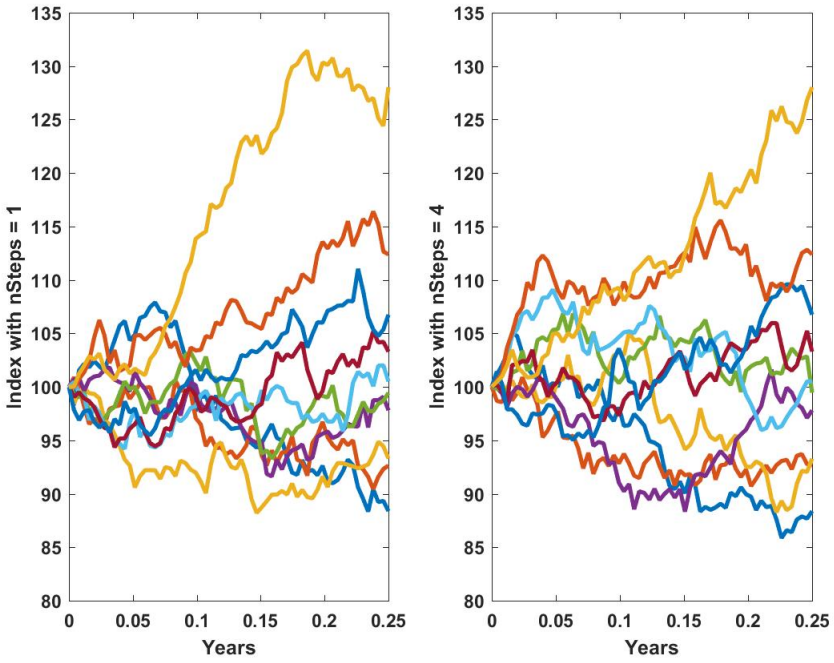


그림 5.5.19. 층화추출법에 의해서 생성된 기하Brown운동의 표본경로

참고 문헌

[1] 최병선 (1997) 회귀분석 *(하)*, 세경사.

[2] 최병선 (2007) 계산재무론 *(IM&F시리즈 10)*, 세경사.

[3] 최병선 (2009) 금융공학 *I: Elements of Financial Engineering (IM&F시리즈 11)*, 세경사.

[4] 최병선 (2013) 금융공학 *II: Black-Scholes Formula (IM&F시리즈 12)*, 세경사.

[5] Acworth, P., M. Broadie, and P. Glasserman (1998) A comparison of some Monte Carlo and quasi Monte Carlo methods for option pricing, pp.1–18 in *Monte Carlo and Quasi-Monte Carlo Methods 1996*, P. Hellekalek, G. Larcher, H. Niederreiter, and P. Zinterhof, eds., Springer-Verlag, Berlin.

[6] Ait-Sahalia, Y. and A. Lo (1998) Nonparametric estimation of state-price densities implicit in financial asset prices, *Journal of Finance*, **53**, 499-47.

[7] Andeerson, T.W. (2003) *An Introduction to Multivariate Statistical Analysis (Third Edition)*, Wiley.

[8] Antanov, I.A., and V.M. Saleev, V.M. (1979) An economic method of computing LP_τ sequences, *USSR Journal of Computational Mathematics and Mathematical Physics* (English translation) **19** 252–256.

[9] Asmussen, S. and P.W. Glynn (2007) *Stochastic Simulation: Algorithms and Analysis*, Spriger.

[10] Brandimarte, P. (2014) *Handbook in Monte Carlo Simulation: Applications in Financial Engineering, Risk Management, and Economics*, Wiley.

[11] Bratley, P., and B. L. Fox (1988) Algorithm 659: Implementing Sobol's quasirandom sequence generator, *ACM Transactions on Mathematical Software*, **14**, 88-100.

[12] Breeden, D. T. and R. H. Litzenberger (1978) Prices of state-contingent claims implicit in option prices, *Journal of Business*, **51**, 621-651.

[13] Caflisch, R.E., W. Morokoff, and A. Owen (1997) Valuation of mortgagebacked securities using Brownian bridges to reduce effective dimension, *Journal of Computational Finance*, **1**, 27–46.

[14] Devroye, L. (1986) *Non-Uniform Random Variate Generation* springer.

[15] Doob, J.L. (1949) Heuristic Approach to the Kolmogorov-Smirnov Theorems, *Annals of Mathematical Statistics* **20**, 393-403.

[16] Donnelly, C. and P. Embrechts (2010) The devil is in the tails: Actuarial mathematics and the subprime mortgage crisis, *ASTIN Bulletin*, **10**, 1-33. Also, available at http://www.math.ethz.ch/∼embrecht/papers.html.

[17] Glasserman, P. (2003) *Monte Carlo Methods in Financial Engineering*, Springer.

[18] Jäckel, P. (2002) *Monte Carlo Methods in Finance*, Wiley.

[19] Joe, S., and F. Y. Kuo (2003) Remark on Algorithm 659: Implementing Sobol's quasirandom sequence generator, *ACM Transactions on Mathematical Software*, **29**, 49-57.

[20] Jöhnk, M.D. (1964) Erzeugung von betaverteilten und gammaverteilten zufallszahlen, *Metrika*, **8**, 5-15.

[21] Jones, S. (2009) The formula that felled Wall St, Financial Times, April 24. Available at http://www.ft.com/intl/cms/s/912d85e8-2d75-11de-9eba-00144feabdc0.

[22] Joy, C., P.P. Boyle, and K.S. Tan (1996) Quasi-Monte Carlo methods in numerical finance, *Management Science*, **42**, 926–938.

[23] Kloeden, P.E. und E. Platen (1992) *The Numerical Solution of Stochastic Differential Equations*, Springer-Verlag.

[24] Kloeden, P.E. und E. Platen (1994) *The Numerical Solution of Stochastic Differential Equations through Computer Experiments*, Springer-Verlag.

[25] Korn, R., E. Korn, and G. Kroisandt *Monte Carlo Methods and Models in Finance and Insurance*, CRC Press.

[26] Kroese, D.P., T. Taimre, and Z.I. Botev (2011) *Handbook of Monte Carlo Methods*, Wiley. Also, refer to http://www.maths.uq.edu.au/ kroese/montecarlohandbook/.

[27] Lilliefors, H. (1967) On the Kolmogorov–Smirnov test for normality with mean and variance unknown, *Journal of the American Statistical Association*, **62**, 399–402.

[28] Lilliefors, H. (1969) On the Kolmogorov–Smirnov test for the exponential distribution with mean unknown, *Journal of the American Statistical Association*, **64**, 387–389.

[29] Marsaglia, G. (1968) Random numbers fall mainly in the planes, *Proc. Natl. Acad. Sci. U.S.A.*, **61**, 25–28. DOI:10.1073/pnas.61.1.25.

[30] Marsaglia, G. (1972) Choosing a point from the surface of a sphere, *Ann. Math. Stat.*, **43**, 645-646.

[31] Marsaglia, G. and T.A. Bray (1964) One-line random number generators and their use in combinations, *Communications of the ACM*, **11**, 737-759.

[32] Marsaglia, G. and W. Tsang (2000) A simple method for generating gamma variables, *Journal ACM Transactions on Mathematical Software*, **26**, 363-372.

[33] Marsaglia, G. and A. Zaman (1991) A new class of random number generators *Annals of Applied Probability*, **1**, 462–480. DOI:10.1214/aoap/1177005878.

[34] Matsumoto, M. and T. Nishimura (1998) Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator, *ACM Trans. on Modeling and Computer Simulation*, **8**, 3-30. DOI:10.1145/272991.272995

[35] Miyazawa, H. and M. Fushimi (2009) An Implementation of a 5-term GFSR Random Number Generator for Parallel Computations, *The Eighth International Symposium on Operations Research and Its Applications (ISORA' 09)*, 448–452.

[36] Niederreiter, H. (1988) Low-discrepancy and low-dispersion Sequences, *Journal of Number Theory*, **30**, 51-70.

[37] Niederreiter, H. (1992) *Random Number Generation and Quasi-Monte Carlo Methods*, Society for Industrial and Applied Mathematics.

[38] Ninomiya, S., and S. Tezuka (1996) Toward real-time pricing of complex financial derivatives, *Applied Mathematical Finance*, **3**, 1–20.

[39] Papageorgiou, A., and J. Traub (1996) Beating Monte Carlo, *Risk*, **9**, 63–65.

[40] Paskov, S.H. and Traub, J.F. (1995) Faster evaluation of financial derivatives. *J. Portfolio Management* **22**, 113-120.

[41] Paskov, S. (1997) New methodologies for valuing derivatives, pp.545–582 in *Mathematics of Derivative Securities*, S. Pliska and M. Dempster, eds., Cambridge University Press.

[42] Ross, S. (1995) *Stochastic Processes (Second Ed)*, Wiley.

[43] Salmon, F. (2009) Recipe for Disaster: The Formula That Killed Wall Street, *Wired Magazine*, **17**, Also, ailable at http://www.wired.com/techbiz/it/magazine/17-03/wp_quant.

[44] Sobol, I. M. (1967) Distribution of points in a cube and approximate evaluation of integrals, *Zh. Vych. Mat. Mat. Fiz.*, **7**, 784-802 (in Russian), *U.S.S.R Comput. Maths. Math. Phys.*, **7**, 86-112 (in English).

[45] Yan, J. (2007) Enjoy the Joy of Copulas: With a Package copula, *Journal of Statistical Software* **21**, 1-21.

찾아보기

저자 소개

최 병 선 (崔 秉 善)

서울대학교 수학과 졸업 (이학사)

미국 Stanford대학교 대학원 졸업 (경제학 석사, 통계학 석사,

Ph. D. in Statistics <major> and Economics <minor>)

연세대학교 상경대학 교수 역임

현재 서울대학교 경제학부 교수 (재무경제학 담당)

E-mail bschoi12@snu.ac.kr

금융공학 IV: Monte Carlo Methods for Finance and Economics

2016년 10월 16일 제2판 1쇄 인쇄

2016년 10월 31일 제2판 1쇄 발행

저 자 **최 병 선**

발행처 **김 구 재 단**

서울특별시 서대문구 충정로9길 10-10 3F (충정로2가)

등 록 2014. 8. 29 제25100-2014-000058호

전 화 02-3146-6923

팩 스 02-312-2675

E-mail smny76@kimkoo.org

I S B N 979 - 11 - 956420 - 4 - 5 93410 비매품

이 도서의 국립중앙도서관 출판예정도서목록(CIP)은 서지정보유통지원시스템 홈페이지 (http://seoji.nl.go.kr)와 국가자료공동목록시스템(http://www.nl.go.kr/kolisnet)에서 이용하실 수 있습니다. (CIP 제어번호 : CIP2016015882)

최병선 교수 Creative Commons Book List

1. (제2판) SAS를 이용한 현대통계학 (이성백교수와 공저)

   http://s-space.snu.ac.kr/handle/10371/94393
2. 행렬의 대각화와 Jordan표준형 (이성백교수와 공저)

   http://s-space.snu.ac.kr/handle/10371/94394
3. Fourier 해석 입문

   http://s-space.snu.ac.kr/handle/10371/94413
4. Lebesgue적분 입문

   http://s-space.snu.ac.kr/handle/10371/94414
5. Wavelet 해석

   http://s-space.snu.ac.kr/handle/10371/94415
6. 회귀분석(상)

   http://s-space.snu.ac.kr/handle/10371/94433
7. 다변량시계열분석

   http://s-space.snu.ac.kr/handle/10371/94434
8. 이산형 재무모형의 수리적 배경

   http://s-space.snu.ac.kr/handle/10371/94455
9. 회귀분석(하)

   http://s-space.snu.ac.kr/handle/10371/94456
10. 단변량시계열분석

    http://s-space.snu.ac.kr/handle/10371/94457