Homework Monte Carlo methods

Student: Hongxing NIU; Matricule ULB: 000342366

1. Monte Carlo methods

Consider the ratio of integrals,

(a). Use **rejection sampling** to generate data X from $f_X(x)$ and estimate $\theta = r_1 = EX$ by $\widehat{\theta_1} = \bar{X}$

PROCEDURE

(1). Generate a random number $X_r = randn * stdev + mu$ from a normal distribution

(2). Compute $f_{XoverMgx} = \frac{1}{1+(\frac{X_r-a}{b})^2}$

(3). Generate a uniform random number $U = rand$

(4). If $U \leq f_{XoverMgx} = \frac{1}{1+(\frac{X_r-a}{b})^2}$ accept $X = X_r$, else reject and go to step 1

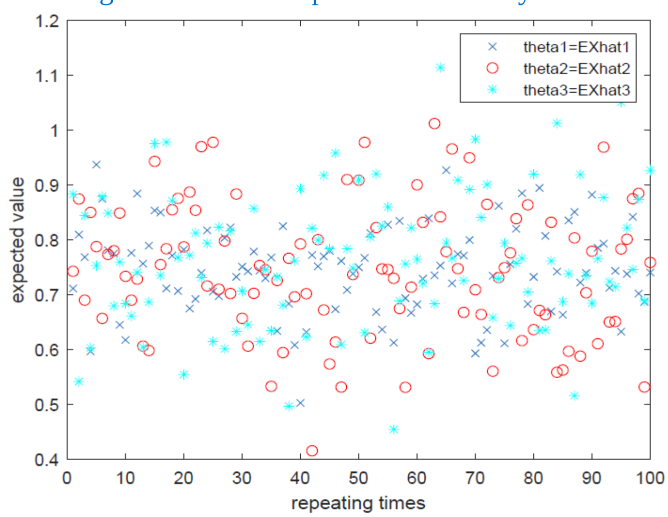
(b). Generate normal data $Z \sim N(\mu, \sigma^2)$ and estimate EX by $\widehat{\theta_2} = \frac{\overline{Z \cdot I_Z}}{\overline{I_Z}}$ with

$I_Z = \frac{1}{b\pi[1+(\frac{Z-a}{b})^2]}$

(c). Generate Cauchy data $T \sim Cauchy(a, b)$ and estimate EX by $\widehat{\theta_3} = \frac{\overline{T \cdot I_T}}{\overline{I_T}}$ with

$I_T = \exp\left[\frac{-(T-\mu)^2}{2\sigma^2}\right] / \sqrt{2\pi}\sigma$

The simulation results using above three samplers are shown by the following figure.



Here, we took n = 2000 samples and repeated 100 times for each of the three samplers; other parameters were a = -3, b = 2, stdev = 4, mu = 5.

In Matlab,
$mean(\widehat{\theta_1}) = mean(EXhat1) = 0.7404;$
$var(\widehat{\theta_1}) = var(EXhat1{,}0) = 0.0057;$

$mean(\widehat{\theta_2}) = mean(EXhat2) = 0.7463;$

$var(\widehat{\theta_2}) = var(EXhat2{,}0) = 0.0185;$
$mean(\widehat{\theta_3}) = mean(EXhat3) = 0.7412;$
$var(\widehat{\theta_3}) = var(EXhat3{,}0) = 0.0171.$

Sample	Size	Variance
1	100	0.0057
2	100	0.0185
3	100	0.0171

- Perform Levene's Test for homogeneity/equality of Variances, i.e, whether $var(\widehat{\theta_1}) = var(\widehat{\theta_2}) = var(\widehat{\theta_3})$

The test statistic is

$$W = \frac{(N-k)\sum_{i=1}^{k} N_i\,(\bar{Z}_{i.} - \bar{Z}_{..})^2}{(k-1)\sum_{i=1}^{k}\sum_{j=1}^{N_i}(\bar{Z}_{ij} - \bar{Z}_{i.})^2}$$

where *Ni* is the sample size of the *i*th group, and *k* is the number of groups.

In the Levene's test, the data are transforming to $Z_{ij} = |Y_{ij} - \bar{Y}_{i.}|$ and uses the *F* distribution performing an one-way ANOVA using Z as the dependent variable (Brownlee, 1965; Miller, 1986)].

In our case, k=3, N=3*100=300, Ni=100,

The Matlab code for Levene's Test for Equality of Variances is shown below,

```
C=(sum(Y(:,1)))^2/length(Y(:,1)); %correction term.
SST=sum(Y(:,1).^2)-C; %total sum of squares.
dfT=length(Y(:,1))-1; %total degrees of freedom.

indice=Y(:,2);
for i=1:k
    Ye=find(indice==i);
    eval(['A' num2str(i) '=Y(Ye,1);']);
end

A=[];
for i=1:k
    eval(['x =((sum(A' num2str(i) ').^2)/length(A' num2str(i) '));']);
    A=[A,x];
end

SSA=sum(A)-C; %sample sum of squares.
dfA=k-1; %sample degrees of freedom.
SSE=SST-SSA; %error sum of squares.
```

```
SSE=SST-SSA; %error sum of squares.
dfE=dfT-dfA; %error degrees of freedom.
MSA=SSA/dfA; %sample mean squares.
MSE=SSE/dfE; %error mean squares.
F=MSA/MSE; %F-statistic.
v1=dfA;df1=v1;
v2=dfE;df2=v2;

P = 1 - fcdf(F,v1,v2);  %probability associated to the F-statistic.

fprintf('Levene''s Test for Equality of Variances F=%3.4f, df1=%2i, df2=%2i\n', F,df1,df2);
fprintf('Probability associated to the F statistic = %3.4f\n', P);

if P >= alpha;
  fprintf('The associated probability for the F test is equal or larger than% 3.2f\n', alpha);
  fprintf('So, the assumption of homoscedasticity was met.\n');
else
  fprintf('The associated probability for the F test is smaller than% 3.2f\n', alpha);
  fprintf('So, the assumption of homoscedasticity was not met.\n');
end
```

The results of Levene's are as follows,

F=6.2700, df1= 2, df2=297;

Probability associated to the F statistic = 0.0022;

The associated probability for the F test is smaller than 0.05;

So, the assumption of homoscedasticity was not met, i.e., the null hypothesis of $var(\widehat{\theta_1}) = var(\widehat{\theta_2}) = var(\widehat{\theta_3})$ is rejected.

2. Markov Chain Monte Carlo sampling

The matlab file *randgammaMCMC.m* implements a Metropolis-Hastings algorithm for the generation of pseudo random variables following a Gamma distribution. It is an alternative to the routine *randgamma.m*, which uses rejection sampling. The file *testrandgammaMCMC.m* compares both generators.

(a). Knowing this, explain how to arrive at the lines of code in *randgammaMCMC.m*

PROCEDURE – METROPOLIS-HASTINGS SAMPLER

(1). Initialize the chain to *X=zeros(m,1)* and set *Y = randexp(1, r, lambda)*, *X(1) = sum(Y)*, where *m=sample size=10,000*, *r = floor(alfa)*. Set *k=2*.

(2). Generate a candidate point *X(k) = sum(Y)* from a proposal distribution *Y = randexp(1, r, lambda)*. Note, the sum of *r* exponential (*lambda*) random variables is a random variable following an Erlang/Gamma (*r*, *lambda*) distribution, *i.e,* $q(x|y) = \frac{x^{\alpha-1}\lambda^r e^{-\lambda x}}{(r-1)!}$.

(3). Generate *U=rand* from a uniform distribution.

(4). If $U \le \frac{f_X(x(k))\cdot q\left(x(k-1)\middle|x(k)\right)}{f_X(x(k-1))\cdot q\left(x(k)\middle|x(k-1)\right)} = [\frac{(x(k))}{(x(k-1))}]^{\alpha-r}$, then accept *X(k)*, else keep the value of *X(k-1)*.

(5). Set $k=2+1$ and repeat steps (2) through (5) until $k \leq m$.

In Matlab, above procedure is implemented by

```
X = zeros(mn,1);

r = floor(alfa);
ralfa = alfa - r;
Y = randexp(1,r,lambda);  % why take r sample
X(1) = sum(Y);

for k=2:mn,
    reject = true;
    while reject,
        Y = randexp(1,r,lambda); X(k) = sum(Y);
        V = (X(k)/X(k-1))^ralfa;
        U = rand;
        if U<V, reject = false; end
    end
end
ans = reshape(X,m,n);
```

(b). Run the file *testrandgammaMCMC.m* and explain in one or two sentences what you see. What could be a drawback of *randgammaMCMC.m* (MCMC sampling) compared to *randgamma.m* (rejection sampling).

Below figure compares the random variables by the method of rejection sampling and by the Metropolis-Hastings algorithm (MH). The MH is an MCMC technique that draws samples from a probability distribution where direct sampling is difficult.

The *cons* of MH are that (i) the simulations $x(k)$ are correlated, hence less informative than i.i.d. simulations; (ii) the validation of the method is only asymptotic, hence there is an approximation in considering $x(k)$ for a fixed k as a realization of f_X . The *pros* of rejection sampling are that there is no approximation in the method: the outcome is truly an i.i.d. sample from f_X.