18.S096 Problem Set 6 Spring 2018
Due Date: 4/13/2018
Where: On Stellar, prior to 11:59pm

Collaboration on homework is encouraged, but you will benefit from independent effort to solve the problems before discussing them with other people. **You must write your solution in your own words. List all your collaborators.**

Problem 1. Monte-Carlo Option Pricing With Non-Gaussian Shocks

Create an R function $MCPrice2()$, an extended version of the Option Pricing Monte Carlo function $MCPrice()$ with an additional argument, $densfun$, that specifies the density for computing independent, normalized shocks. Include the following distributions as options for the argument:

- Laplace distribution
- Student's t distribution with 5 degrees of freedom
- Gaussian distribution (as currently implemented)

Review the R scripts OptionPlots1.r and OptionMC1.r in RProject3. In the current version of $MCPrice()$, the following code snippet creates a matrix of i.i.d. realizations from the standard Gaussian distribution, i.e., $Normal(mean = 0, sd = 1)$.

```
# Select independent, normalized Gaussian shocks
epsilon <- matrix(rnorm(Nt*Np), ncol=Np)
```

To conform with option-pricing theory, your function must compute the matrix *epsilon* as i.i.d. realizations from normalized distributions (i.e., zero mean and unit standard deviation).

1(a) For a $Laplace(location = a, scale = b)$ distribution, determine the *normalized* parameter values (a, b) which make the distribution mean equal 0 and the distribution standard deviation equal 1.

Create the r function $rlaplace()$ which generates pseudo-random realizations from this normalized Laplace distribution.

1(b) For a sample realization $X \sim t_5$, from a t distribution with $df = 5$ degrees of freedom, determine the mean and standard deviation of X

$$\mu = E[X] \text{ and } \sigma = \sqrt{Var[X]}$$

Give explicit formula for σ. The normalized transformation

$$Z = (X - \mu)/\sigma.$$

is a normalized t_5 random variable.

Create the r function $rt5normalized()$ which generates pseudo-random realizations from this normalized t_5 distribution.

1(c) Using the functions $rlaplace()$ and $rt5normalized()$ from parts (a) and (b), create $MCPrice2()$, a revision of the function $MCPrice()$ which allows different distributions for the normalized shocks.

Hint: You may find it helpful to review the syntax of the function $fitdistr()$ in the library $MASS$ to see how a function can be structured to handle different distributions with an argument

$$densfun = [rnorm|rlaplace|rt5normalized]$$

1(d) Apply $MCPrice2()$ for each of the 3 distribution options to compute Monte-Carlo option prices of calls and puts for the same case used in RProject3: call/put for at-the-money options $S0 = K = 100$, maturity $T = 1$ year with annual volatility $sigma = 0.3$

```
## One-year horizon (T=1)
#  Daily increments (Nt=252)
#  10,000 paths
#  At-the-money calls/puts (S0=K=100)
#
S0 <- 100; K <- 100; T <- 1; rf <- 0.03; sigma <- 0.3;
Nt <- 252; Np <- 1e4; dt=T/Nt
set.seed(1)
MCprice(S0,K,rf,T,sigma,Nt,Np)
```

Problem 2. Monte Carlo Option Pricing (Continued)

2(a) In problem 1, the Monte Carlo distribution of option prices depends on the distribution of terminal path values for the dynamics of the asset price under the risk-neutral probability measure. For T=1 year maturity, the simulation uses $Nt = 252$ daily increments. Should the central limit theorem (CLT) apply in approximating the distribution of the terminal path values? If so, would this lead to obtaining different Monte-Carlo prices for the different distributions in problem 1?

2(b) Repeat the comparisons of 1(d), changing the maturity to 1 month: apply $T = 1/12$ years. Maintain the daily increments for the simulation and set $Nt = 21(= 252/12)$ days (typical market days in a month).

2(c) Based on your answers to problem 1 and to parts (a) and (b), comment on the sensitivity of the Monte Carlo option price to

- the choice of shock distribution
- the maturity T of the option
- the number of increments (Nt) in the Monte Carlo paths

Problem 3. Stationary and Ergodic Distributions

The $AR(p = 1)$ Gaussian process is given by:

$$X_{t+1} = \alpha_0 + \alpha_1 X_t + \epsilon_{t+1}$$

where $\{\epsilon_t\}$ are i.i.d. $Normal(0, \sigma^2)$.

3(a) Derive the density of the conditional distribution of $X_{t+1} \mid X_t = x_t$

3(b) Suppose $|\alpha_1| < 1$ and that $x_t \sim f(x)$, where $f(\cdot)$ is the density of a $Normal(\mu_0, \sigma_0)$ distribution with

$$\mu_0 = \alpha_0/(1 - \alpha_1)$$
$$\sigma_0^2 = \sigma^2/(1 - \alpha_1^2)$$

Show that the marginal distribution of x_{t+1} has the same density $f(\cdot)$, i.e., it is the *stationary* distribution of the process.

3(c) Set $X_0 = 0.$ and simulate a realization/path $X_t, t \leq T = 1000$ with $\alpha_0 = 1.$ and $\alpha_1 = 0.95$ and $\sigma = 1$. Construct two plots: the time series plot of the sample path and the histogram of the sample $\{x_t\}$. Check the fit of the stationary distribution $Normal(\mu_0, \sigma_0)$ to the histogram. (Superpose the curve of the staitonary distribution on the histogram)

3(d) **Sensitivity to starting value.** Repeat (b) twice: with $X0 = 20$, and with $X0 = 100$. Explain the influence of the starting value.

3(e) **Decreasing sensitivity to starting value with longer series.**

Repeat (d) with $T = 10,000$.

3(f) For estimating the stationary distribution with the sample distribution of a single-path's values, comment on the value of excluding path values during an initial *burn*-in period. Should the length of an effective burn-in period depend on the starting value? Suggest strategies for choosing initial values and specifying the burn-in period of a sample path.

Problem 4. Accept-Reject Sampling and Metropolis-Hasting Sampling

In the R package library *mcsm*, the R demo script *Chapter.6.r* details examples applying the Metropolis-Hastings algoithm; see RProject4 and the script file *Chapter.6_rev1.r*.

4(a) The section comparing Accept-Reject and Metropolis-Hasting algorithms for generating gamma random variables is reproduced below.

```
Section 6.2.2, comparison of gamma generators

a=4.85;nsim=10000;
X1=X2=array(0,dim=c(nsim,1))           #AR & MH
X1[1]=X2[1]=rgamma(1,a,rate=1)               #initialize the chain
for (i in 2:nsim){
    Y=rgamma(1,floor(a),rate=floor(a)/a)    #candidate
    rhoAR=(exp(1)*Y*exp(-Y/a)/a)^(a-floor(a))
    rhoMH=(dgamma(Y,a,rate=1)/dgamma(X2[i-1],a,rate=1))/(dgamma(Y,floor(a),
    rate=floor(a)/a)/dgamma(X2[i-1],floor(a),rate=floor(a)/a))
    rhoMH=min(rhoMH,1)
    X1[i]=Y*(runif(1)<rhoAR)                          #accepted values
    X2[i]=X2[i-1] + (Y-X2[i-1])*(runif(1)<rhoMH)
}
X1=X1[X1!=0]                     #The AR sample
par(mfrow=c(2,2),mar=c(4,4,2,2))
hist(X1,col="grey",nclas=125,freq=FALSE,xlab="",main="Accept-Reject",xlim=c(0,15))
curve(dgamma(x, a, rate=1),lwd=2,add=TRUE)
hist(X2[2500:nsim],nclas=125,col="grey",freq=FALSE,xlab="",main="Metropolis-Hastings",x
curve(dgamma(x, a, rate=1),lwd=2,add=TRUE)
acf(X1,lag.max=50,lwd=2,col="red")              #Accept-Reject
acf(X2[2500:nsim],lag.max=50,lwd=2,col="blue")  #Metropolis-Hastings
```

Revise the code to compare gamma generators for shape $a = 1.75$. In addition to reproducing the corresponding plots of the demo script, compare the cumulative "rejection" rates of the Accept-Reject and Metropolis-Hastings algorithms. (Note: the variable $X1$ in the code initially has zeros for rejected values, and $X2$ has rejected values when $X2[t] == X2[t-1]$)

4(b) The section comparing the Metropolis-Hasting algorithm for generating $Beta(a = 2.7, b = 6.3)$ random variables to the direct generation method is reproduced below. Revise the script to generate $Beta(a = 0.5, b = 0.5)$ random variables.

```
a=2.7; b=6.3;  # initial values
# initial values
nsim=5000
```

```
X=rep(runif(1),nsim)  # initialize the chain
for (i in 2:nsim){
   Y=runif(1)
   rho=dbeta(Y,a,b)/dbeta(X[i-1],a,b)
   X[i]=X[i-1] + (Y-X[i-1])*(runif(1)<rho)
   }
#X11(h=3.5);m
plot(4500:4800,X[4500:4800],ty="l",lwd=2,xlab="Iterations",ylab="X")

ks.test(jitter(X),rbeta(5000,a,b))

par(mfrow=c(1,2))
hist(X,nclass=150,col="grey",main="Metropolis-Hastings",fre=FALSE)
curve(dbeta(x,a,b),col="sienna",lwd=2,add=TRUE)
hist(rbeta(5000,a,b),nclass=150,col="grey",main="Direct Generation",fre=FALSE)
curve(dbeta(x,a,b),col="sienna",lwd=2,add=TRUE)
```

4(c) For the $Beta(a = b = 0.5)$ distribution in (b), it is not possible to apply the Accept-Reject method. Explain why.

4(d) The function *hastings*() in the library *mcsm* demonstrates using the Metropolis-Hastings algorithm as originally given by Hastings (1970). Simulated Markov-Chain paths whose ergodic distributions are $Normal(0, 1)$ are generated using a random walk where the step-sizes are a symmetric uniform distributions of width $2a$. The function applies three cases of a: $0.1, 1., 10.$.

Revise the function to demonstrate the algorithm with values $0.5, 1., 5$.

In R you can edit the function by writing

```
library("mcsm")
hastings2<-hastings
# Then edit hastings2 directly
fix(hastings2) # change the vector a
# Execute the revised function
hastings2()
```

- Explain why the acceptance rate decreases with a
- Is it better to always use a candidate distribution with a higher acceptance rate?
- Explain why the ACF function has higher values the smaller the value of a, the half-width of the uniform candidate density.