Computer Practical: Metropolis-Hastings
Model Answers

Model code in Python

```
from __future__ import division
from scipy import*
from scipy import linalg
import pylab
import sys
sys.path.append("/home/ludger/lib/python2.6/site-packages/")
import statistics

#Task 1: Evaluate the density of a bivariate distribution at a single point
def density(x,mu,sigma):
        inv_sigma=linalg.inv(sigma)
        x_minus_mu=x-mu
        return exp(-0.5*dot(dot(transpose(x_minus_mu),inv_sigma),x_minus_mu))/(2*pi*sqrt(
linalg.det(sigma) ) )

mu=array([0,0])
sigma=array([[4,1],[1,4]])

#Task 2: Metropolis Hastings

#Set the standard deviation of the proposal
sigma_prop=2.5
#Set the desired sample size
n=1000
#Set the starting values
x=array([0,0])
accepted_n=0
f=density(x,mu,sigma)
X1=[x[0]]
X2=[x[1]]

for i in xrange(1,n):
        x_0=x[0]+random.normal(0,sigma_prop)
        x_1=x[1]+random.normal(0,sigma_prop)
        new_x=array([x_0,x_1])
        new_f=density(new_x,mu,sigma)
        if (random.random()<(new_f/f)):
                accepted_n+=1
                x=new_x
                f=new_f
        X1.append(x[0])
        X2.append(x[1])

#Proportion of accepted values
print "The proportion of accepted values is", accepted_n/n

#Task 3: Diagnostics

#Sample plots for both values
pylab.figure(0)
pylab.plot(X1,'b')
pylab.title("Sample path of X_1")
pylab.figure(1)
pylab.plot(X2,'r')
pylab.title("Sample path of X_2")

#Cumulative averages
```

```
X1_cummean = cumsum( X1 ) / ( 1 + arange( len( X1 )))
X2_cummean = cumsum( X2 ) / ( 1 + arange( len( X1 )))
pylab.figure( 2 )
pylab.plot( X1_cummean,"b" )
pylab.title("Empirical mean of X_1")

pylab.figure( 3 )
pylab.plot( X2_cummean,"r" )
pylab.title("Empirical mean of X_2")
pylab.show()

#Autocorrelation
X1_sd = sqrt( var( X1 ) )
X2_sd = sqrt( var( X2 ) )
X1_autocorr = statistics.correlation( X1[1: ], X1[:-1])
X2_autocorr = statistics.correlation( X2[1: ], X2[:-1])
print "The autocorrelation of X_1 is", X1_autocorr
print "The autocorrelation of X_2 is", X2_autocorr

#Effective sample size
X1_ess = n * ( 1 - X1_autocorr ) / ( 1 + X1_autocorr )
X2_ess = n * ( 1 - X2_autocorr ) / ( 1 + X2_autocorr )
print "The effective sample size of X_1 is", X1_ess
print "The effective sample size of X_2 is", X2_ess

#Task 4: Repeat with sigma_prop = 0.1, sigma_prop = 10

#Task 5: Repeat with sigma = array([[4,2.8],[2.8,4]])
```

Model code in R

```
#Task 1
#Evaluates the density of a bivariate normal distribution at a single point
mvdnorm <- function(x, mu, sigma){
        x.minus.mu <- x - mu
                #subtract mu from x
        exp.arg <- -0.5 * sum(x.minus.mu * solve(sigma, x.minus.mu))
                #evaluates the expression inside exp(...)
        return( 1 / (2 * pi * sqrt(det(sigma))) * exp(exp.arg) )
        }

#Task 2
#Metropolis Hastings

#set the mean parameter
mu <- c(0,0)
#set the covarince parameter
sigma <- matrix(c(4,1,1,4), nrow=2)

#set the standard deviation of the proposal
sd.proposal <- 2.5
#set the desired sample size
n <- 1000
x <- matrix(nrow=n, ncol=2)

#set the starting value
cur.x <- c(0,0)
#evaluate the density at the starting value
cur.f <- mvdnorm(cur.x, mu, sigma)

n.accepted <- 0
for(i in 1:n){
        new.x <- cur.x + sd.proposal * rnorm(2)
        new.f <- mvdnorm(new.x, mu, sigma)
        if(runif(1) < new.f/cur.f){
                n.accepted <- n.accepted + 1
                cur.x <- new.x
                cur.f <- new.f
                }
        x[i,] <- cur.x
        }

#proportion of accepted values
n.accepted/n

#Task 3
#look at sample plots of both variables
par(mfrow=c(2,1))
plot(x[,1], type=”l”, xlab=”t”, ylab=”X_1”, main=”Sample␣path␣of␣X_1”)
plot(x[,2], type=”l”, xlab=”t”, ylab=”X_2”, main=”Sample␣path␣of␣X_2”)

#compute cumulative averages
x1_cumulative <- rep(0, times=n)
x2_cumulative <- rep(0, times=n)
for(i in 1:n){
  x1_cumulative[i] <- mean(x[1:i,1])
  x2_cumulative[i] <- mean(x[1:i,2])
}

par(mfrow=c(2,1))
plot(x1_cumulative, type=”l”, xlab=”t”, ylab=”mean␣of␣X_1”, main=”Empirical␣mean␣of␣X_1”)
plot(x2_cumulative, type=”l”, xlab=”t”, ylab=”mean␣of␣X_2”, main=”Empirical␣mean␣of␣X_2”)

#correlations
cor.x1 <- cor(x[-1,1], x[-nrow(x),1])
cor.x2 <- cor(x[-1,2], x[-nrow(x),2])
c(cor.x1, cor.x2)
```

```
#effective sample size
ess.x1 <- n * (1-cor.x1) / (1+cor.x1)
ess.x2 <- n * (1-cor.x2) / (1+cor.x2)
c(ess.x1, ess.x2)

#Task 4
#Change the standard deviation of the proposal, and repeat.
sd.proposal <- 0.1
sd.proposal <- 10

#Task 5
#Now set the covarince parameter as follows, and repeat.
sigma <- matrix(c(4,2.8,2.8,2), nrow=2)
```