MATH 565 Monte Carlo Methods in Finance

Fred J. Hickernell **Fall 2011**

Take-Home Final **Due 2 pm, Thursday, December 8**

Instructions:

i. *This take-home part of the final exam has THREE questions. You may attempt as many as you like. Your score will consist of your best TWO answers for a total possible of* 50 marks.

ii. *You may consult any book, web page, software repository, notes, old tests, or other inanimate object. You may use the m-files on Blackboard. You* may not *consult any other person face-to-face, by telephone, by email, Facebook, Twitter, LinkedIn or by any other means.*

iii. *Show all your work to justify your answers. Submit hard copies of your derivations, programs, output, and explanations to my mail box in E1-208. (Off-site students only may submit by email or Blackboard Dropbox.) Answers without adequate justification will not receive credit.*

iv. *In addition, as a precaution, submit soft copies of your programs to the Blackboard Dropbox. If I have difficulty understanding your computational work, I may look at your programs.*

1. (25 marks)
Consider the problem of approximating

$$\mu = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \cos(x^2 + y^2)\, e^{-(x^2+y^2)/2}\, \mathrm{d}x\, \mathrm{d}y.$$

Use both simple Monte Carlo and scrambled Sobol' sequences to evaluate this integral to absolute error tolerances of $0.02, 0.01, 0.005, 0.002, 0.001,$ and 0.0005.

- Record the time, t, and sample size, n, needed by each method for each tolerance. Which method works better for which tolerances?
- Decide what is the maximum sample size that is feasible for each method for your computing environment. If the sample size required is more than the maximum, then record a failure by the method. Which method(s) fails for which tolerances?
- For simple Monte Carlo, estimate the sample size needed using the Central Limit Theorem approximation.
- For scrambled Sobol' sampling, use $m = 30$ different scramblings of the Sobol' sequence, and estimate the error using the mean of each different scrambling. Starting from an initial sample size, continue to double the sample size for the Sobol' sequences until the error tolerance is met.

```
%% Garbage collection
close all, clear all, format compact

%% Problem 1
tolvec=[0.05 0.02 0.01 0.005 0.002 0.001 0.0005];
for ii=1:length(tolvec);
tol=tolvec(ii); %error tolerance

%% Simple Monte Carlo
n0=1000; %initial sample size
d=2; %dimension
```

```
nmax=1e8/d; %maximum feasible sample size
muhatstr='??? out of memory'; %answer in string form
fun=@(x) (2*pi)*cos(sum(x.*x,2)); %integrand

%Initial sample
tstart=tic;
x=randn(n0,d); %standard Gaussian random vectors
fx=fun(x); %integrand values
stdf0=std(fx); %standard deviation of function values
n=ceil(1.5*(1.96*stdf0/tol)^2);
if n<=nmax;
    x=randn(n,d); %standard Gaussian random vectors
    muhat=mean(fun(x)); %approximation to the integral
    muhatstr=num2str(muhat);
end
tMC=toc(tstart);
disp('By i.i.d. sampling')
disp(['Integral = ' muhatstr ' +/- ' num2str(tol)])
disp(['     samples required = ' num2str(n+n0)])
disp(['     time elapsed = ' num2str(tMC)])

%% Sobol' sampling
tstart=tic;

%Create scrambled Sobol' samples
nrep=30;
nmax=1e8/(d*nrep);
Sob=sobolset(d);
ScrSob=cell(nrep);
for i=1:nrep
    ScrSob{i}=scramble(Sob,'MatousekAffineOwen');
end

%Keep increasing sample size until done
overallmean=0;
n=0;
nnext=2; %initial sample size
sumpc=zeros(1,nrep);
errest=inf;
while errest>tol && n<nmax
    fxpc=zeros(nnext,nrep);
    for i=1:nrep
        fxpc(:,i)=fun(norminv(net(ScrSob{i},nnext)));
    end
    sumpc=sumpc+sum(fxpc,1);
    n=n+nnext;
    meanpc=sumpc/n;
    errest=1.96*std(meanpc)/sqrt(nrep);
    nnext=2*nnext;
```

```
end
muhat=mean(meanpc); %approximation to the integral
tSob=toc(tstart);
disp('By Sobol sampling')
disp(['Integral = ' num2str(muhat) ' +/- ' num2str(tol)])
disp(['     samples required = ' num2str(n*nrep)])
disp(['     time elapsed = ' num2str(tSob)])
disp(' ')
end

By i.i.d. sampling
Integral = 1.2484 +/- 0.05
     samples required = 45812
     time elapsed = 0.023991
By Sobol sampling
Integral = 1.2853 +/- 0.05
     samples required = 15300
     time elapsed = 0.29357

By i.i.d. sampling
Integral = 1.2421 +/- 0.02
     samples required = 279955
     time elapsed = 0.02125
By Sobol sampling
Integral = 1.2551 +/- 0.02
     samples required = 61380
     time elapsed = 0.09722

By i.i.d. sampling
Integral = 1.2619 +/- 0.01
     samples required = 1102487
     time elapsed = 0.073093
By Sobol sampling
Integral = 1.2566 +/- 0.01
     samples required = 122820
     time elapsed = 0.11338

By i.i.d. sampling
Integral = 1.2565 +/- 0.005
     samples required = 4284101
     time elapsed = 0.24531
By Sobol sampling
Integral = 1.2555 +/- 0.005
     samples required = 245700
     time elapsed = 0.13352

By i.i.d. sampling
Integral = 1.2563 +/- 0.002
     samples required = 27798951
     time elapsed = 1.5416
By Sobol sampling
Integral = 1.2572 +/- 0.002
     samples required = 982980
     time elapsed = 0.24516

By i.i.d. sampling
Integral = ??? out of memory +/- 0.001
     samples required = 112290891
     time elapsed = 0.02439
By Sobol sampling
Integral = 1.2571 +/- 0.001
     samples required = 1966020
     time elapsed = 0.40106

By i.i.d. sampling
Integral = ??? out of memory +/- 0.0005
     samples required = 457644002
     time elapsed = 0.00025149
By Sobol sampling
Integral = 1.2568 +/- 0.0005
     samples required = 7864260
     time elapsed = 1.2534
```

Answer: For larger tolerances, in this case 0.01 *or greater, i.i.d. sampling is faster, but for smaller tolerances, Sobol' sampling is faster.*

2. (25 marks)
One kind of low discrepancy set that is related to the linear congruential generator is the integration lattice. The idea is to run a small linear congruential generator to its full period and include the point zero. For this problem we consider an artificially small case in dimension $d = 2$. The integration lattice is defined by

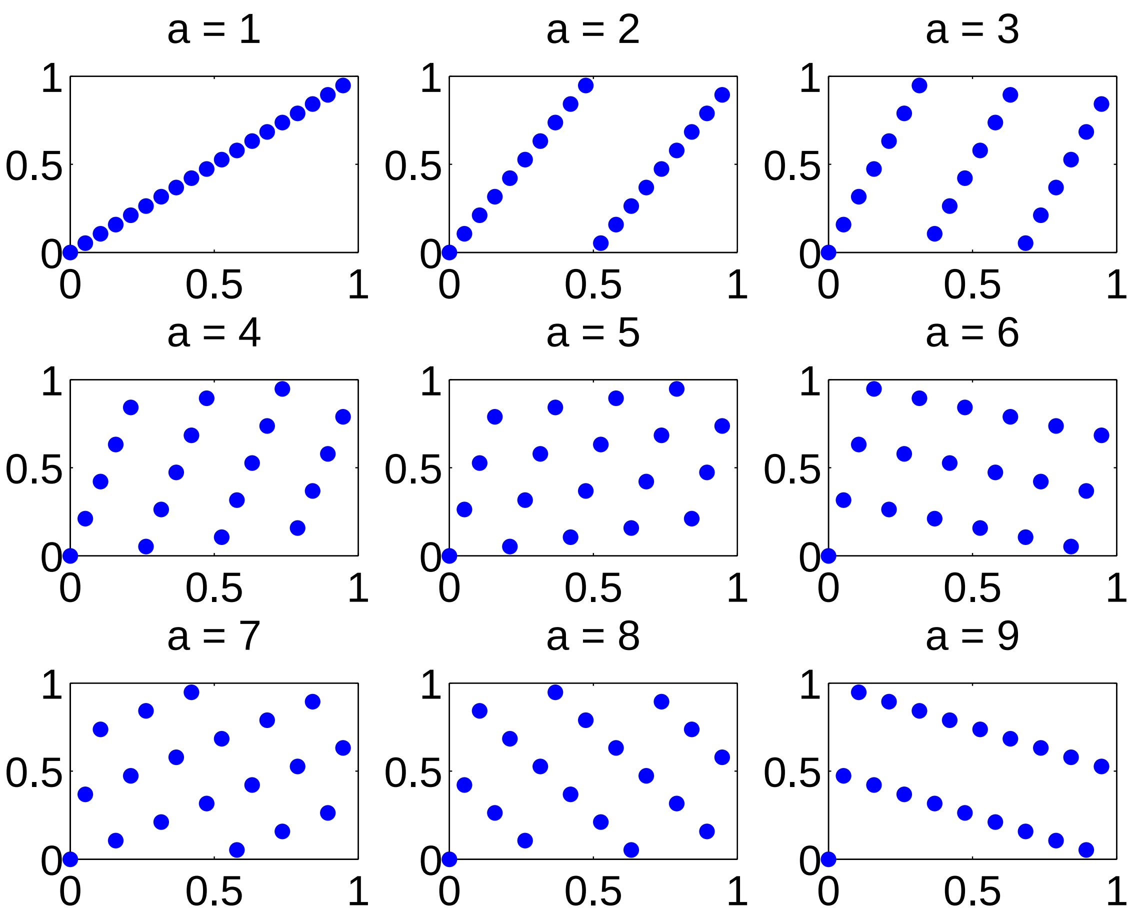
$$\boldsymbol{x}_i = (x_{i1}, x_{i2}) = (1, a)i/n \pmod 1, \quad i = 0, \dots, n-1.$$

It is not necessary that a be a primitive root, but only relatively prime with respect to n. For $n = 19$ find the best generator, a, by visual inspection, and by using either one of the spectral tests or a discrepancy measure (your choice) for dimension 2. Note that by symmetry arguments, you only need to consider $a = 1, \dots, 9$.

```
%% Spectral Test for Integration Lattice
clear all
tic

n=19;
disp(['For n = ' int2str(n)])
na=floor(n/2);
```

```
aposs=(1:na)'; %possible a values
naposs=length(aposs); %number of possible a values
disp('The possible values of the parameter a are:')
disp(aposs')

nplotx=ceil(sqrt(na));
nploty=ceil(na/nplotx);
for ii=1:na
    subplot(nplotx,nploty,ii)
    plot((0:n-1)/n,mod((0:n-1)*ii/n,1),'b.','markersize',20)
    title(['a = ' int2str(ii)])
end
print -depsc aplot.eps

%% Perform the Spectral Test for d=2
nucand1d=(-n:n)'; %1d candidate wave numbers
[nu1cand2d,nu2cand2d]=ndgrid(nucand1d);
nucand2d=[nu1cand2d(:) nu2cand2d(:)];
nucand2d=nucand2d(any(nucand2d~=0,2),:);
dotprod=mod(nucand2d*[ones(1,naposs); aposs'],n);
whdual=dotprod==0;
l1minnorm=zeros(naposs,1);
l2minnorm=l1minnorm;
whl1minnu=zeros(naposs,2);
whl2minnu=whl1minnu;
for j=1:naposs;
    nudual2d=nucand2d(whdual(:,j),:);
    norm1dual=sum(abs(nudual2d),2);
    norm2dual=sqrt(sum(nudual2d.*nudual2d,2));
    [l1minnorm(j),whmin]=min(norm1dual);
    whl1minnu(j,:)=nudual2d(whmin(1),:);
    [l2minnorm(j),whmin]=min(norm2dual);
    whl2minnu(j,:)=nudual2d(whmin(1),:);
end
disp('The l1 spectral test is')
disp(l1minnorm')
disp('  attained at wavenumbers')
disp(whl1minnu')
besta=aposs(l1minnorm==max(l1minnorm));
disp(['The best values of a according to this test are ' int2str(besta') ])
disp(' ')
disp('The l2 spectral test is')
disp(l2minnorm')
disp('  attained at wavenumbers')
disp(whl2minnu')
besta=aposs(l2minnorm==max(l2minnorm));
disp(' ')
disp(['The best values of a according to this test are ' int2str(besta') ])
toc
```

```
For n = 19
The possible values of the parameter a are:
     1      2      3      4      5      6      7      8      9
The l1 spectral test is
     2      3      4      5      5      4      5      5      3
  attained at wavenumbers
     1      2      3      4      1     -1      2     -3     -1
    -1     -1     -1     -1     -4     -3     -3     -2     -2
The best values of a according to this test are 4  5  7  8

The l2 spectral test is
  Columns 1 through 7
    1.4142    2.2361    3.1623    4.1231    4.1231    3.1623    3.6056
  Columns 8 through 9
    3.6056    2.2361
  attained at wavenumbers
     1      2      3      4      1     -1      2     -3     -1
    -1     -1     -1     -1     -4     -3     -3     -2     -2
The best values of a according to this test are 4  5

Elapsed time is 0.872689 seconds.
```



Answer: The best values of a are 4 or 5.

3. (25 marks)
   On November, 17, 2011, Google shares closed at \$600.87 per share. The price of an American put option expiring on January 18, 2014 with a strike price of \$550 was \$83.20. Assume a geometric Brownian motion model for the price of Google, an interest rate of $r = 1\%$, a volatility of $\sigma = 40\%$ and $d = 32$ time steps. Does the Monte Carlo approximation to the put price match the market price to the nearest dollar? If not, change one or more of the parameters, r, σ, or d to match the market price to the nearest dollar. Explain why you changed the parameter(s) that you decided to change.

   Answer: The time to expiry for this option is 2.17 years. Using the `OptionPrice.m` *program one gets price of \$103, which is higher than the market price. Changing the number of time steps has a negligible effect, and the interest rate should not be changed too much because 1% is close to the market rate. The biggest influence is the volatility. By decreasing that to about 34% we get a price close to the market price.*

```
Using 100000 asset price samples based on a
   discrete geometric Brownian motion model of the asset
   with sampling method:
     independent and identically distributed
For an initial asset price of $600.87
   a strike price of $550.00
   2.17 years to maturity
   an interest rate of 1.00%
   a volatility of 40.00%:
For American put options monitored 32 times
The put price is $103.80940.7002
Compared to the GBM European call price of $165.5380
   and the GBM European put price of $102.8616
This computation took 0.53008 seconds

Using 100000 asset price samples based on a
   discrete geometric Brownian motion model of the asset
   with sampling method:
     independent and identically distributed
For an initial asset price of $600.87
   a strike price of $550.00
   2.17 years to maturity
   an interest rate of 1.00%
   a volatility of 40.00%:
For American put options monitored 64 times
The put price is $103.49300.6940
Compared to the GBM European call price of $165.5380
   and the GBM European put price of $102.8616
This computation took 0.80017 seconds
```

```
Using 100000 asset price samples based on a
   discrete geometric Brownian motion model of the asset
   with sampling method:
     independent and identically distributed
For an initial asset price of $600.87
   a strike price of $550.00
   2.17 years to maturity
   an interest rate of 1.00%
   a volatility of 33.50%:
For American put options monitored 32 times
The put price is $83.14040.5966
Compared to the GBM European call price of $145.0312
   and the GBM European put price of $82.3547
This computation took 0.38611 seconds
```