**EE511 project 8 Zhang Fan USC ID 1417 68 5115**

# Question 1

## Method

According to the title of statement, we need implement Monte Carlo estimate and variance, stratified sampling and important sampling to compare the sample variance and the quality of result of (a) and (b).

For stratified sampling,

* generate random number two dimension x and put them into the equation.
* Calculate the mean and variance of equation for the estimated equation

For important sampling

* get and g(x), which we can know its pdf function
* With inverse transform method to obtain the sample
* Put them into h(x)/f(x), the result can be obtained

## Code for (a)

clear;

N=1000;

x=rand(2,N);

g = exp(sum(5\*abs(x-5)));

y=rand(1,N).\*max(g);

sum=0;

for i=1:N

if(g(i)>y(i))

sum=sum+1;

end

end

area=sum/N\*max(g);

disp([area 2\*std(g)/sqrt(N)])

%---------------Stratified sampling:

clear

g = @(x)exp(sum(5\*abs(x-5)));

N = 1000; X = g(rand(2,N)); % Simple MC

disp([mean(X) 2\*std(X)/sqrt(N)])

K = 10; Nij = N/K^2; % Stratified

for i = 1:K

for j = 1:K

XS = g([i-1+rand(1,Nij);j-1+rand(1,Nij)]/K);

XSb(i,j) = mean(XS); SS(i,j) = var(XS);

end

end, SST = mean(mean(SS/N));

disp([mean(mean(XSb)) 2\*sqrt(SST) ])

%--------------------Importance sampling

clear;

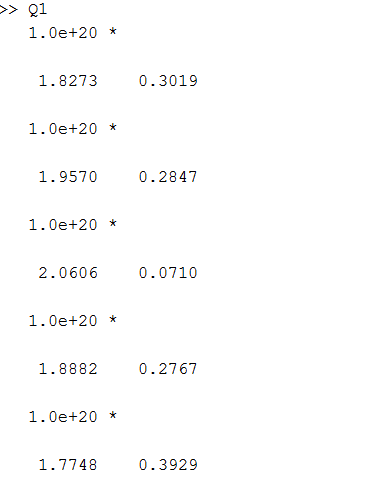
N = 1000; U = rand(2,N); T = exp(sum(5\*abs(U-5)));

disp( [mean(T) 2\*std(T)/sqrt(N)]) % simple MC

e = exp(1); X = log(1+(e-1)\*U);

T = (e-1)^2\*exp(sum(5\*abs(X-5))-sum(X));

disp( [mean(T) 2\*std(T)/sqrt(N)])



## Code for b

clear;

N=1000000;

x=2\*rand(2,N)-1;

g = cos(pi+sum(5\*x));

y=(rand(1,N)\*2-1).\*max(g);

sum1=0;

sum2=0;

for i=1:N

if(g(i)>=0)

if(g(i)>y(i)&&y(i)>0)

sum1=sum1+1;

end

elseif(g(i)<y(i)&&y(i)<0)

sum2=sum2+1;

end

end

area=(sum1-sum2)/N\*2\*(max(g)-min(g))

disp([area\*2 2\*std(g)/sqrt(N)])

%---------------Stratified sampling:

clear

g = @(x)cos(pi+sum(5\*x));

N = 100000; X = g(2\*rand(2,N)-1); % Simple MC

disp([mean(X)\*2\*2 2\*std(X)/sqrt(N)])

K = 10; Nij = N/K^2; % Stratified

for i = 1:K

for j = 1:K

XS = g([-1+2\*(i-1)/K+2\*(rand(1,Nij)/K);-1+2\*(j-1)/K+2\*(rand(1,Nij)/K)]);

XSb(i,j) = mean(XS); SS(i,j) = var(XS);

end

end, SST = mean(mean(SS/N));

disp([mean(mean(XSb))\*2\*2 2\*sqrt(SST) ])

%--------------------Importance sampling

clear;

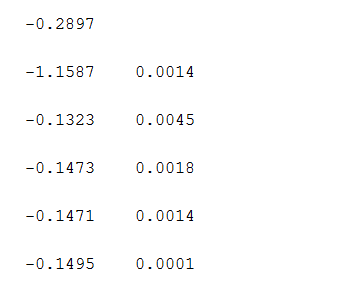
N = 1000000; U = -1+2\*rand(2,N); T = cos(pi+sum(5\*U));;

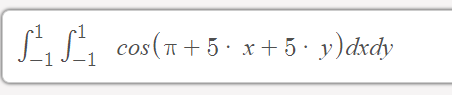
disp( [mean(T)\*2\*2 2\*std(T)/sqrt(N)]) % simple MC

e = exp(1/2); X = (acos(U)-pi)/5;

T = (cos(pi+sum(5\*X)))/10/(-e);

disp( [mean(T)\*2\*2 2\*std(T)/sqrt(N)])



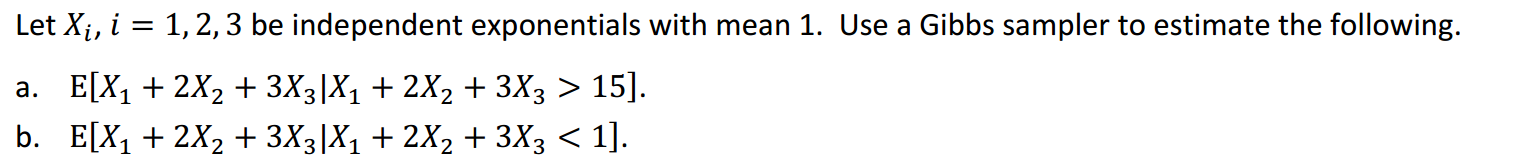


=-0.14

## Comment and comparison

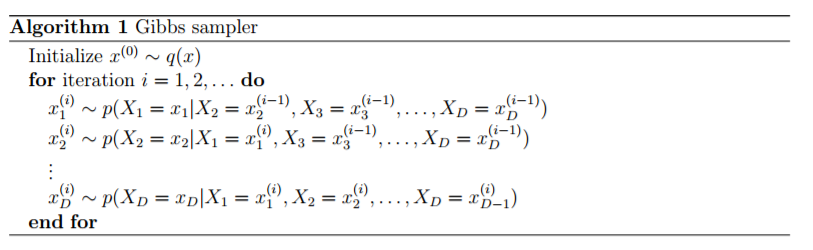
It is obvious that the result with stratified sampling and important sampling will have lower variance, and their expectation are accurate. Therefore, compared to straight Monte Carlo, stratified sampling and important sampling are preferred.

# Question 2



## Analysis

The main idea is to change only one sample (for example x1, or x2 or x3) and calculate their expectation.



## Code

%======gibbs method============

clear

N=1000

x=exprnd(1,1,N);

X123=[0,0,0]';

mean=0;

for i=1:N

if(sum(X123)<1)

mean=mean+sum(X123);

end

X123(mod(i,3)+1)=x(i)%change one element one time

end

mean=mean/N;

%====reject and accept method to verify the result=================

clear

N=1000;

x=exprnd(1,3,N);

mean=0;

for i=1:N

if(sum(x(:,i))<1)

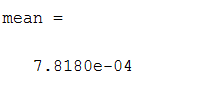
mean=mean+sum(x(:,i));

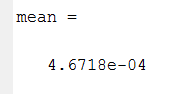
end

end

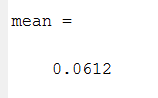
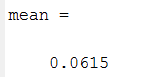
mean=mean/N

## result for part (a)

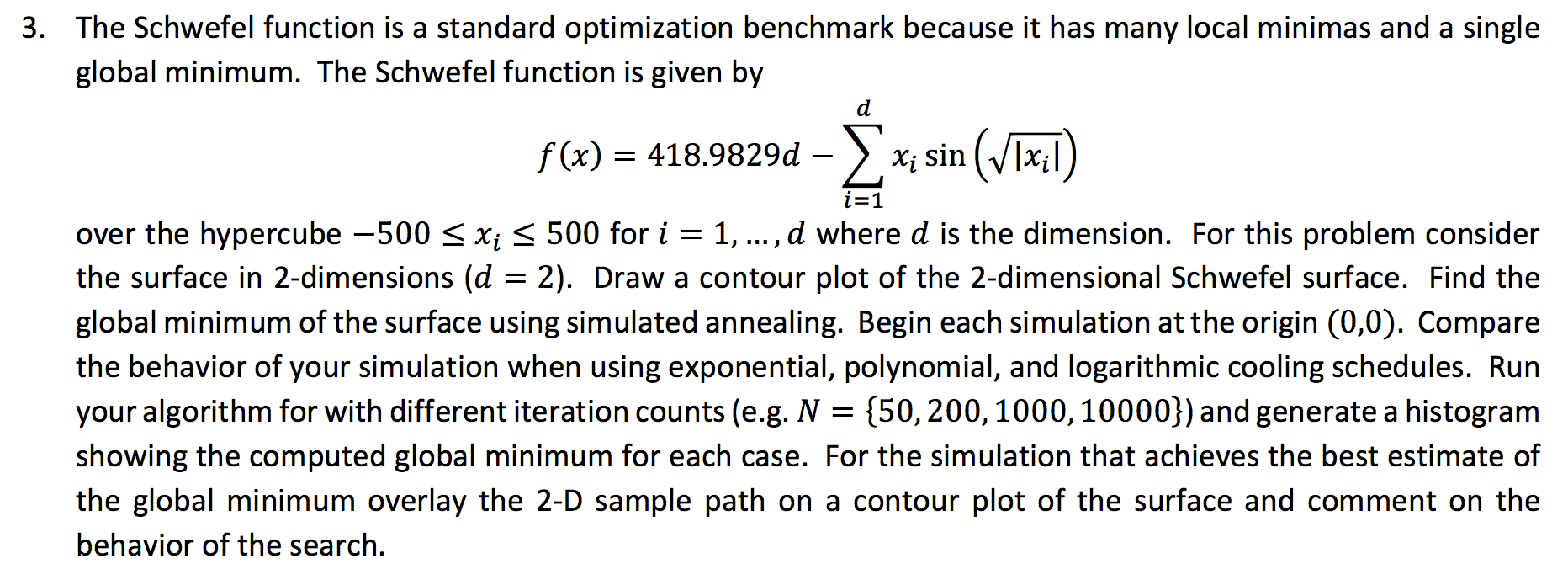
reject and accept method 

gibbs method 

## result for part (b)

reject and accept methodgibbs method

# Question 3



## Analysis:

* Step 1, choose initial guess for solution.
* Step 2, generate candidate by sampling from normal.
* Step 3, computer α by evaluating the function at the old and new point.
* Step 4, make the jump or stay.
* Step 5, cooling schedule.
* Step 6, repeat.

## Code for contour

x = linspace(-500,500); % Generate a row vector of 100 linearly equally spaced points between -512 and 512

y = linspace(-500,500); % Generate a row vector of 100 linearly equally spaced points between -512 and 512

[X,Y] = meshgrid(x,y); % Obtain 100x100 pairs of points in matrix form from vectors x and y

Z = 418.9829\*2-X\*sin(sqrt(abs(X))+Y\*sin(sqrt(abs(Y))));% Compute f(x,y) in matrix form

% Compute f(x,y) in matrix form

figure(1)

contour(X,Y,Z); % Plot a contour plot

colorbar;

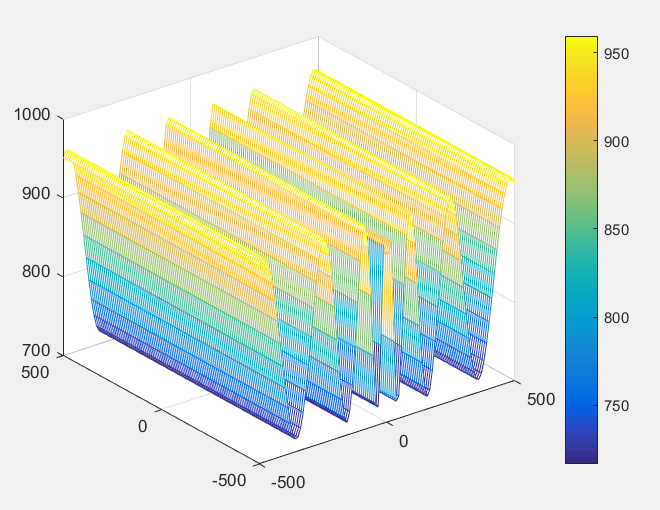
figure(2)

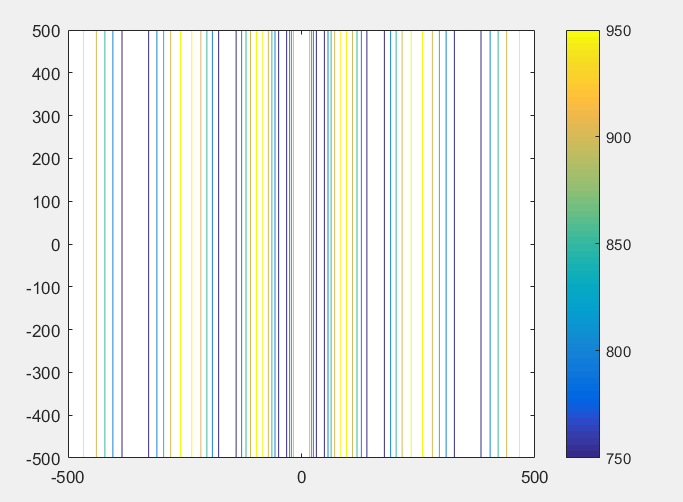
mesh(X,Y,Z); % Plot a mesh plot

colorbar;

xlim([-500 500]);

ylim([-500 500]);





## Code for find global minimum

function [BestX,BestY]=SimulateAnnealing1

clear;

clc;

%axil;

ResultX=0;

ResultY=0;

XMAX= 4;

YMAX = 4;

%initialize

MarkovLength = 10000;

DecayScale = 0.95;

StepFactor = 0.02;

Temperature=30;

Tolerance = 1e-8;

AcceptPoints = 0.0;

rnd =rand;

% choose point randomly

PreX = -XMAX \* rand ;

PreY = -YMAX \* rand;

PreBestX = PreX;

PreBestY = PreY;

PreX = -XMAX \* rand ;

PreY = -YMAX \* rand;

%begin at (0,0)

BestX = 0;

BestY = 0;

% itiration

mm=abs( ObjectFunction( BestX,BestY)-ObjectFunction (PreBestX, PreBestY));

itirationNumber=1000;%initial number

for circle=1:itirationNumber

%Temperature=DecayScale\*Temperature;%polymial

%Temperature=Temperature/(log(itirationNumber+1));%log

Temperature=exp(-itirationNumber\*(itirationNumber)^(1/DecayScale));%

AcceptPoints = 0.0;

%

for i=0:MarkovLength:1

% 1)

p=0;

while p==0

NextX = PreX + StepFactor\*XMAX\*(rand-0.5);

NextY = PreY + StepFactor\*YMAX\*(rand-0.5);

if p== (~(NextX >= -XMAX && NextX <= XMAX && NextY >= -YMAX && NextY <= YMAX))

p=1;

end

end

% 2)

if (ObjectFunction(BestX,BestY) > ObjectFunction(NextX,NextY))

%

PreBestX =BestX;

PreBestY = BestY;

ResultX(circle)=NextX;

ResultY(circle)=NextY;

BestX=NextX;

BestY=NextY;

end

% 3) Metropolis

if( ObjectFunction(PreX,PreY) - ObjectFunction(NextX,NextY) > 0 )

%//

PreX=NextX;

PreY=NextY;

AcceptPoints=AcceptPoints+1;

else

changer = -1 \* ( ObjectFunction(NextX,NextY) - ObjectFunction(PreX,PreY) ) / Temperature ;

rnd=rand;

p1=exp(changer);

double (p1);

if p1 > rand %//

PreX=NextX;

PreY=NextY;

AcceptPoints=AcceptPoints+1;

end

end

end

mm=abs( ObjectFunction(BestX,BestY)-ObjectFunction (PreBestX, PreBestY));

end

BestX

BestY

ObjectFunction(BestX, BestY)

(reference from internetand revised )

