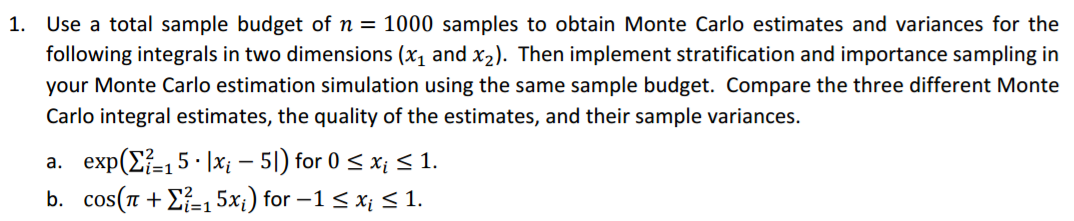
**Rajasekar Raja**

**Last 3 digits of USC ID- 494**

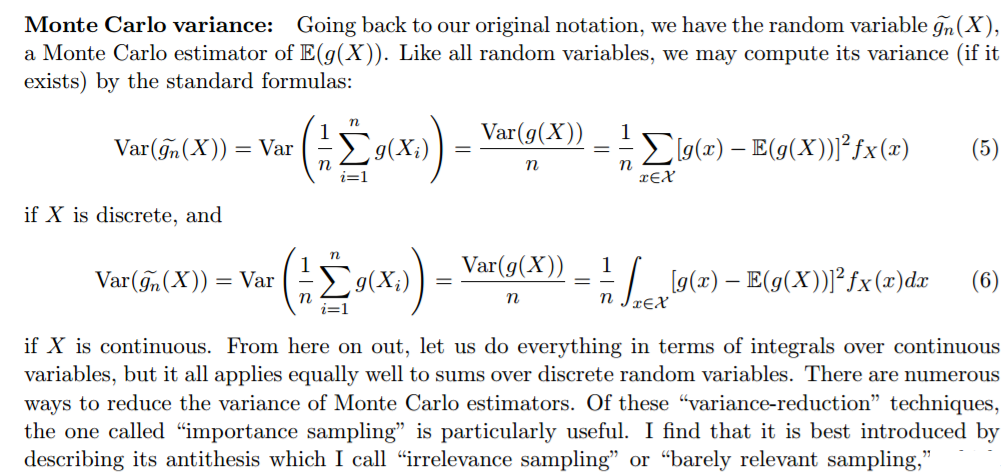
**Tuesday-10 to 10.50AM**

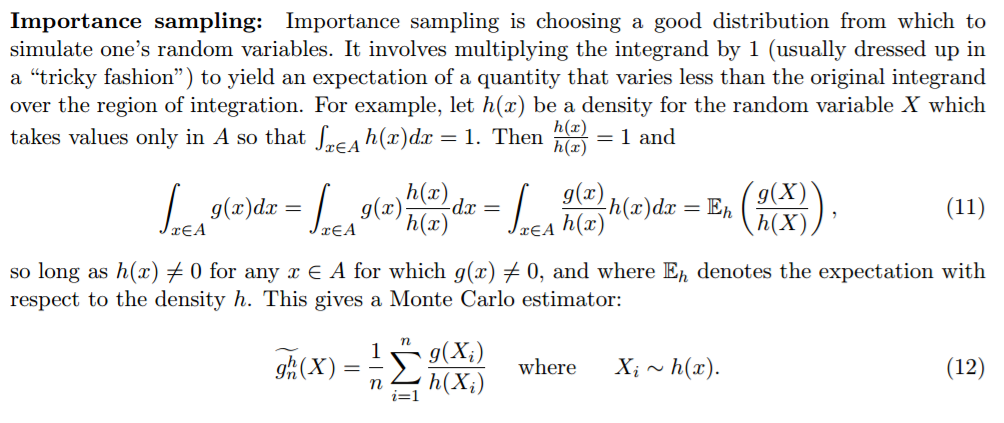
Project #8 – Markov Chain Monte Carlo

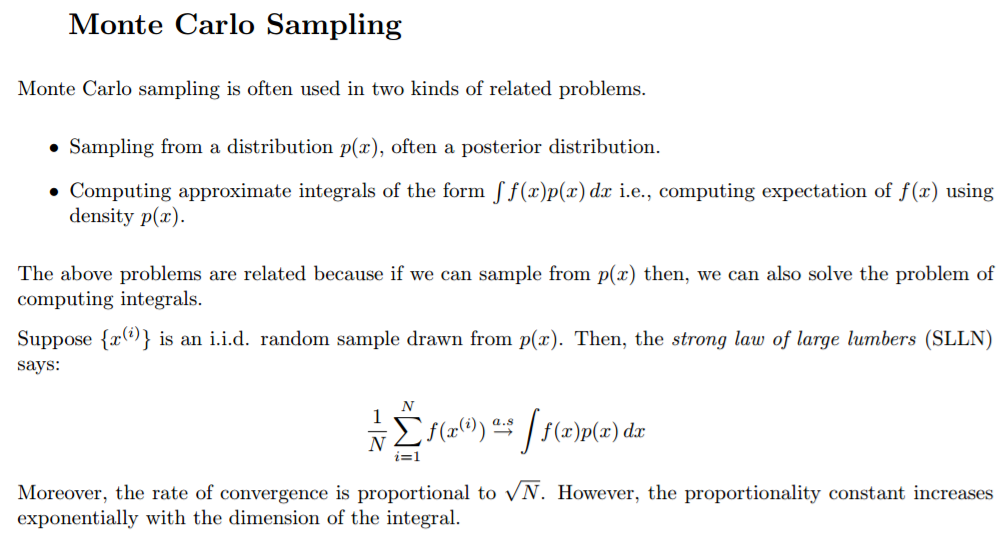
**Concept behind this problem:**

The question requires the knowledge of ***Stratified, Monte Carlo and Importance sampling***.

Stratified Sampling**:** It refers to a type of sampling method . With stratified sampling, the researcher divides the population into separate groups, called strata. Then, a probability sample (simple random sample) is drawn from each group. Stratified sampling has several advantages over simple random sampling.It may be possible to reduce the sample size required to achieve a given precision or it may be possible to increase the precision with the same sample size.







**Reference:**

**[1] Piazza**

**[2]** [**https://people.eecs.berkeley.edu/~jordan/courses/260-spring10/lectures/lecture17.pdf**](https://people.eecs.berkeley.edu/~jordan/courses/260-spring10/lectures/lecture17.pdf)

**Code & Code Description:**

* Firstly, since the question requires us to Estimate mean and variance using simple Monte Carlo, importance and stratified sampling, we create a generic code and reuse it.
* In the first part upon user input, we generate two sets of random numbers and use it to calculate the function as per the question.
* We then calculate the Mean and Variance of the function.

% Estimating mean and variance using simple Monte Carlo

r\_number1 = rand(1,No\_of\_samples);

r\_number2 = rand(1,No\_of\_samples);

**part-a**

function = @( r\_number1, r\_number2)exp(5.\*abs(r\_number1-5)+5.\*abs(r\_number2-5));

**Part-b (a=1 and b=-1 – limits of x1 and x2)**

function =(b-a)\*(b-a)\*cos(pi + 5\*(a + (b-a)\*X12) + 5\*(a + (b-a)\*X22));

display(['Mean ',num2str(mean(X)),' and Variance ',num2str(2\*std(X)/sqrt(No\_of\_samples)),' using simple Monte Carlo']);

* For stratified sampling, we divide the total sample into few subgroups (40 in this case).
* Once we have the subgroups, we pick samples from each subgroups and then apply it to the function.

%Estimating mean and variance using simple Stratified sampling

Number\_of\_subgroups = 40;

Nij = No\_of\_samples/ Number\_of\_subgroups;

for i = 1: Number\_of\_subgroups

for j = 1: Number\_of\_subgroups

r\_number1 = i-1+rand(1,Nij))/ Number\_of\_subgroups;

r\_number2 = j-1+rand(1,Nij))/ Number\_of\_subgroups;

**part-a**

function = @(r\_number1, r\_number2)exp(5.\*abs(r\_number1-5)+5.\*abs(r\_number2-5))

**part-b**

function = @( r\_number1, r\_number2)cos(pi + 5.\* r\_number1 + 5.\* r\_number);

end end

SST = mean(mean(var(function)/No\_of\_samples));

display(['Mean ',num2str(mean(mean(mean(function)))),' and Variance ',num2str(2\*sqrt(SST)),' using Stratified Sampling']);

* In importance sampling, we sample the distribution that overweight from the importance distribution. By doing that we can observe the expectation.
* So, we generate two variables and derive few random numbers as mentioned below within the limits of the function.
* We then apply log function with exp(1) and use it in the function provided in the question.

%Estimating mean and variance using simple Importance sampling

r\_num1 = (b-a).\*rand(1,Nij)+ a;

r\_num2 = (b-a).\*rand(1,Nij)+ a;

r\_num1 = log(1+(exp(1)-1)\*r\_num1);

r\_num2 = log(1+(exp(1)-1)\*r\_num2);

**part-a**

function = exp(1)-1)^2\*exp(5.\*abs(r\_num1-5) + 5.\*abs(r\_num2-5)- (X1+X2);

**part-b**

function = (exp(1)-1)^2\*cos(pi + 5.\*r\_num1 + 5.\*r\_num2-(r\_num1+r\_num2);

display(['Mean ',num2str(mean(function)),' and Variance ',num2str(2\*std(function)/sqrt(No\_of\_samples)),' using simple Monte Carlo']);

* Finally, we calculate the theoretical value by directly using the integral function in matlab with limits as given in the question.

**Part-a**

funtion = @(0,1) exp(5.\*abs(0-5) + 5.\*abs(1-5));

theo\_val = integral2 (@(0,1)fun(0,1),0,1,0,1);

**Part-b**

funtion = @(-1,1) cos(-1 + 5\*-1 + 5\*1);

theo\_val = integral2 (@(-1,1)fun(-1,1),-1,1,-1,1);

disp(['Theoretical integral value',num2str(theo\_val)]);

**Results and Theoretical Values:**

>> ee511\_p8\_q1(1000)

Mean 1.898180517341201e+20 and Variance 2.955542809916657e+19 using simple Monte Carlo

Mean 2.043497318918254e+20 and Variance 3.254616482895911e+18 using Stratified Sampling

Mean 1.864318314948884e+20 and Variance 4.797866761085293e+19 using Importance Sampling

Theoretical integral value 2.046028949028753e+20

Mean -0.18308 and Variance 0.17559 using simple Monte Carlo

Mean -0.0015591 and Variance 0.0092094 using simple Monte Carlo

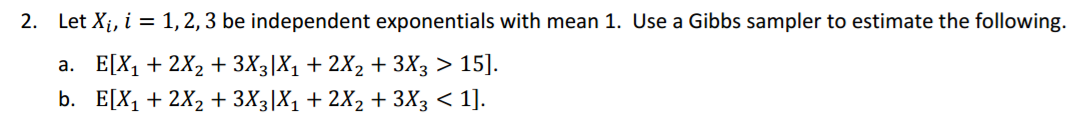
Mean -0.1711002 and Variance 0.45344328 using simple Monte Carlo

Theoretical integral value -0.14713

**Observations:**

* We observe that the three different Monte Carlo integral estimates, the quality of the estimates, and their sample variances are almost same matching the theoretical value.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

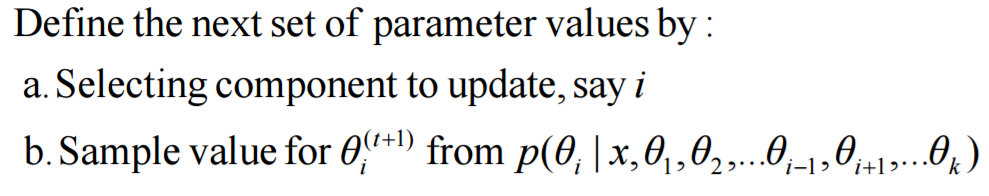


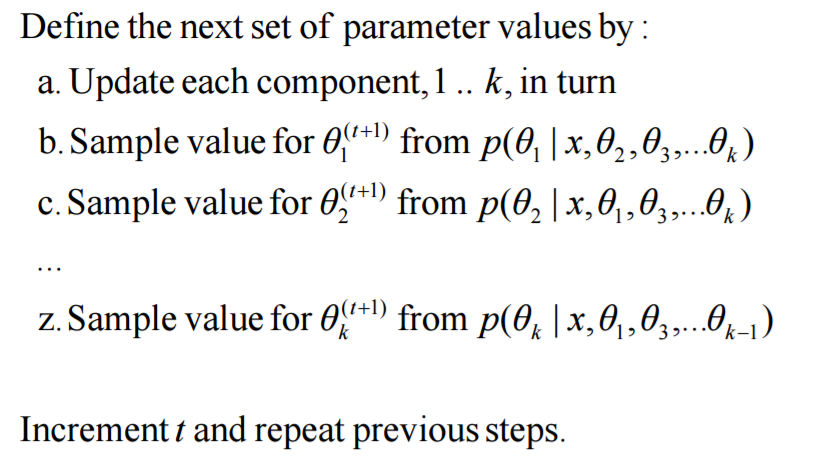
**Concept behind this problem:**

The problem uses the concept of **Gibbs sampling**.

Gibbs sampler is a Markov chain Monte Carlo algorithm for obtaining a sequence of observations which are approximated from a specified multivariate probability distribution, when direct sampling is difficult. This sequence can be used to approximate the joint distribution (e.g., to generate a histogram of the distribution); to approximate the marginal distribution of one of the variables, or some subset of the variables.







**Reference:**

**[1] Piazza**

**[2]** [**http://csg.sph.umich.edu/abecasis/class/815.23.pdf**](http://csg.sph.umich.edu/abecasis/class/815.23.pdf)

**Code and Description:**

* Here a generic function is made to handle both the part of questions by iterating it in a loop.
* One other point here is that the gibbs sampling is used when the random variables are dependent.
* Once the user inputs the number of samples, we calculate the individual terms expectation given the function of X1,X2,X3 for each term and then multiply in the end.
* Usage of log function is to simplify the multiplication .

function [ ] = gibbs\_sampling(No\_of\_samples)

num\_rand\_var = 3;

rand\_var\_vec = 4\*ones(1,num\_rand\_var);

given\_mean=1;

for iteration = 1:2

if (part ==1)

constant = 15;

else

constant = 1;

end

for iter = 1 : No\_of\_samples

rand\_var\_index = ceil(num\_rand\_var\*rand);

S = sum(rand\_var\_vec) - rand\_var\_index\*rand\_var\_vec(rand\_var\_index);

if (part ==1)

rand\_var\_vec(rand\_var\_index) = max(constant-S,0)-log(rand)/given\_mean;

else

rand\_var\_vec(rand\_var\_index) = log(rand)/given\_mean- max(constant-S,0);

end

sum\_rand\_var(iter) = S + rand\_var\_vec(rand\_var\_index);

end

if (part ==1)

disp(['E[X1+2X2+3X3|X1+2X2+3X3>15]: The Mean of the estimate is ',num2str(mean(sum\_rand\_var))]);

else

disp(['E[X1+2X2+3X3|X1+2X2+3X3<1]: The Mean of the estimate is ',num2str(mean(sum\_rand\_var))]);

end end end

**Results and Plots:**

>> ee511\_p8\_q2(1000)

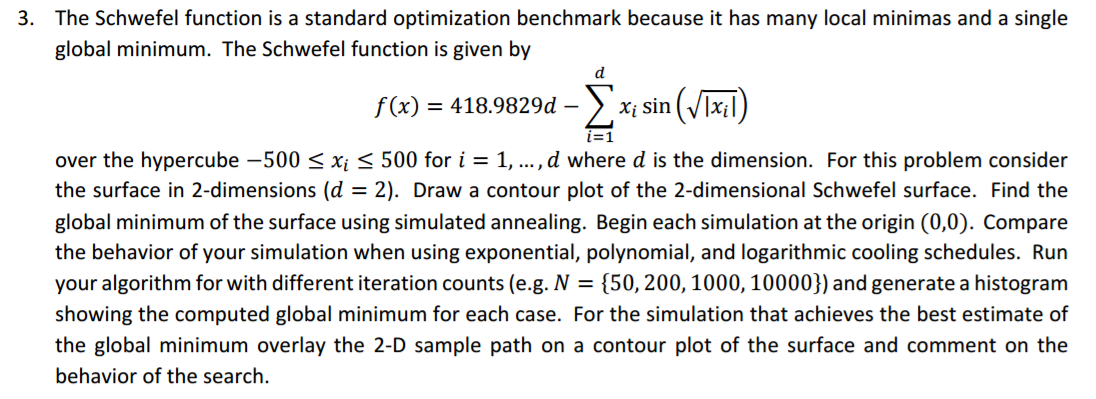
E[X1+2X2+3X3|X1+2X2+3X3>15]: The Mean of the estimate is 20.8933

E[X1+2X2+3X3|X1+2X2+3X3<1]: The Mean of the estimate is 6.67371

**Observations:**

* Exponential Random variable follows the memoryless property i.e. E[X|X>c]= E[X] +c.

Here in part a, it is ~21 and part b it is ~7 which matches the observed value via simulation.

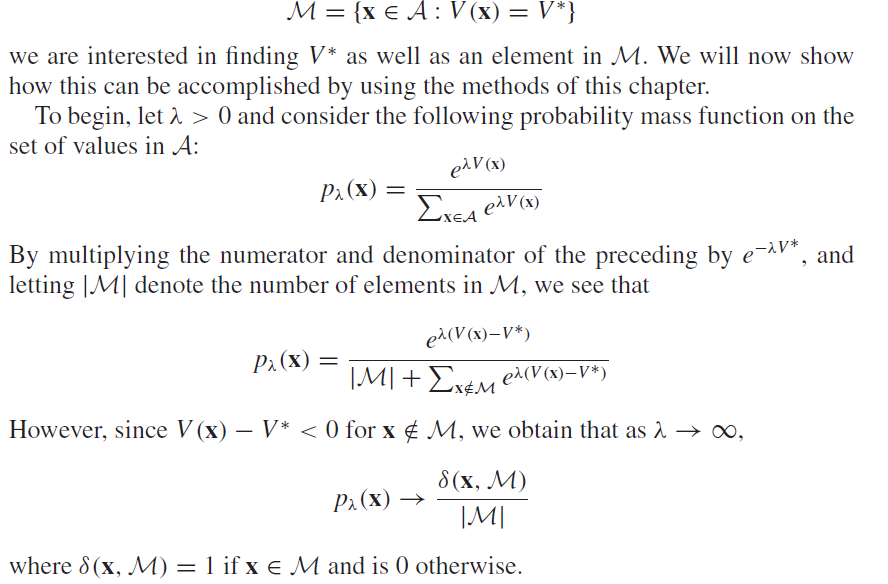


**Concept behind this problem:**

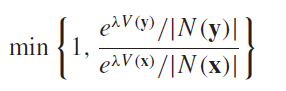
The problem uses the concept of ***Simulated annealing***.

Simulated annealing is a probabilistic technique for approximating the global optimum of a given function. Specifically, it is a metaheuristic to approximate global optimization in a large search space. It is often used when the search space is discrete (e.g., all tours that visit a given set of cities).

Let A be a finite set of vectors and let V(x) be a nonnegative function defined on x ∈ A, and suppose that we are interested in finding its maximal value and at least one argument at which the maximal value is attained. That is, letting V∗ = max(x∈A) V(**x**)



Hence, if we let λ be large and generate a Markov chain whose limiting distribution is pλ(x), then most of the mass of this limiting distribution will be concentrated on points inM. An approach that is often useful in defining such a chain is to introduce the concept of neighboring vectors and then use a Hastings– Metropolis algorithm. For instance, we could say that the two vectors x ∈ A and y ∈ A are neighbors if they differ in only a single coordinate or if one can be obtained from the other by interchanging two of its components. We could then let the target next state from x be equally likely to be any of its neighbors, and if the neighbor y is chosen, then the next state becomes y with probability



**Reference:**

**[1] Piazza**

**[2] Simulation by** [**Sheldon**](http://mccormickml.com/2014/08/04/gaussian-mixture-models-tutorial-and-matlab-code/) **Ross**

**Code & Code Description:**

* Here we choose the linspace function of matlab to create equally spaced points between the limits of X given in the problem (-500 to 500).
* We then create a meshgrid and use the value in the function provided in the question.

function [] = ee511\_p8\_q3(no\_of\_samples)

min\_limit = -500;

max\_limit = 500;

equ\_spaced\_pts1 = linspace(min\_limit,max\_limit); %Row vector of 100 linearly equally spaced points between -512 and 512

equ\_spaced\_pts2 = equ\_spaced\_pts1;

[equ\_spaced\_pts1,equ\_spaced\_pts2] = meshgrid(equ\_spaced\_pts1,equ\_spaced\_pts2);

func = 418.9829\*2 - equ\_spaced\_pts1.\*sin(sqrt(abs(equ\_spaced\_pts1))) - equ\_spaced\_pts2.\*sin(sqrt(abs(equ\_spaced\_pts2)));

* We then use the normrnd variable to calculate the parameters which are substituted to find the counter plot as shown below

for iter = 1 :no\_of\_samples

value1(iter+1) = matrix\_x(iter) +normrnd(0,10);

value2(iter+1) =matrix\_y(iter) + normrnd(0,10);

val1 = 418.9829\*2 - matrix\_x(iter)\*sin(sqrt(abs(matrix\_x(iter)))) - matrix\_y(iter)\*sin(sqrt(abs(matrix\_y(iter))));

val2 = 418.9829\*2 - value1(iter+1)\*sin(sqrt(abs(value1(iter+1)))) - value2(iter+1)\*sin(sqrt(abs(value2(iter+1))));

alpha = exp(val1 -val2)/ matrix\_Z(iter);

if ((val2 <= val1) || (rand(1)<alpha))

matrix\_x(iter+1) = value1(iter+1);

matrix\_y(iter+1) = value2(iter+1);

else

matrix\_x(iter+1)= matrix\_x(iter);

matrix\_y(iter+1) = matrix\_y(iter);

end

matrix\_Z(iter+1) = 100/log(iter+1);

end

value1(no\_of\_samples)

value2(no\_of\_samples)

**Graphs and Plots**

figure(1);

contour(equ\_spaced\_pts1,equ\_spaced\_pts2,func);

colorbar;

figure(2)

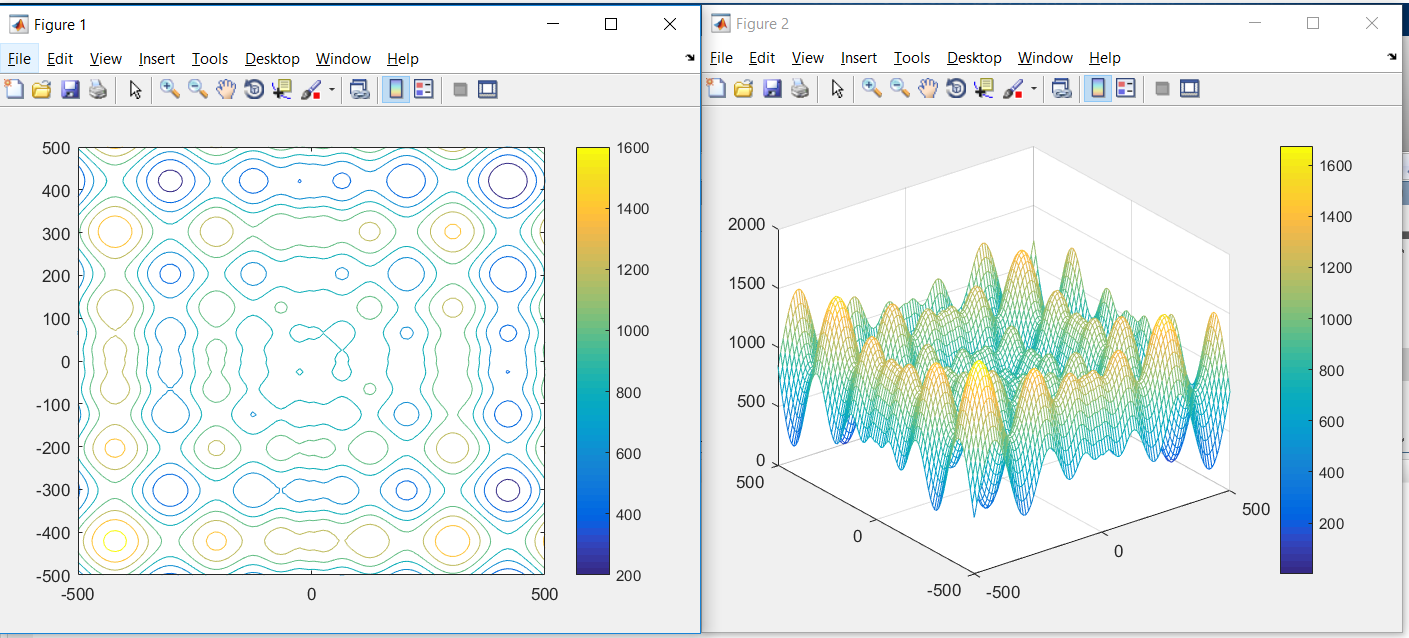
mesh(equ\_spaced\_pts1,equ\_spaced\_pts2,func);

colorbar;

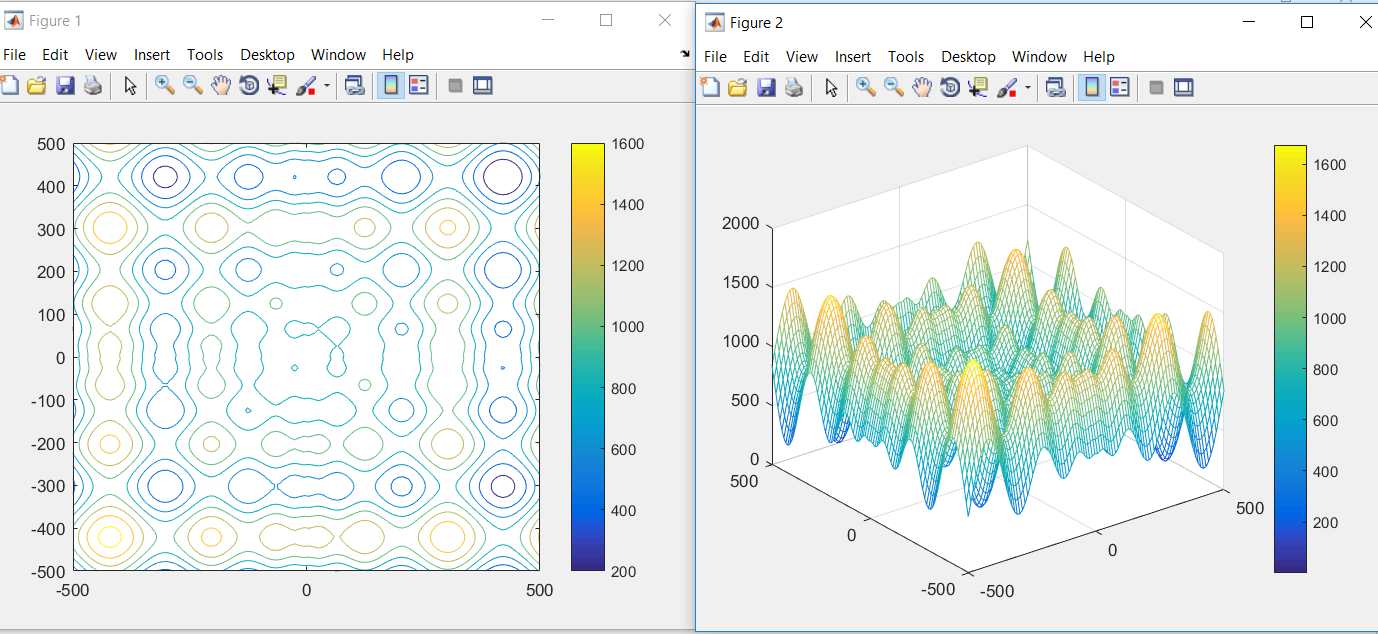
xlim([-500 500]);

ylim([-500 500]);

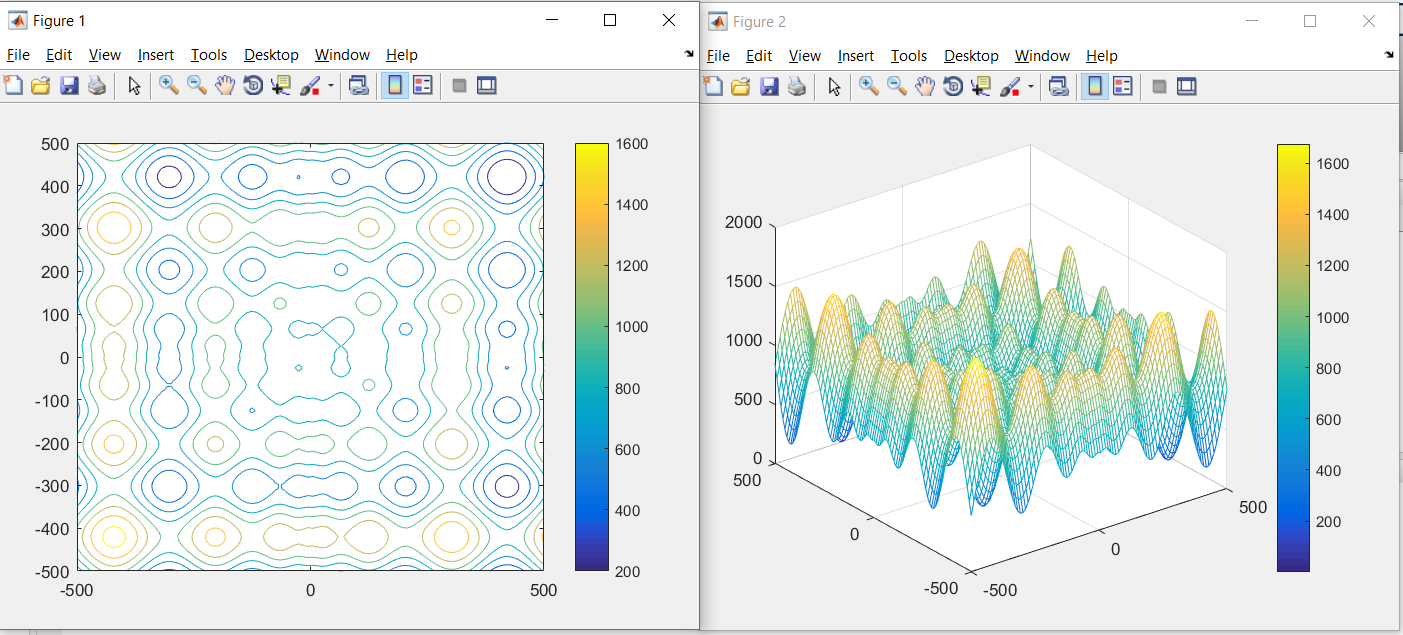
The counter and 3D plot for 50 iterations



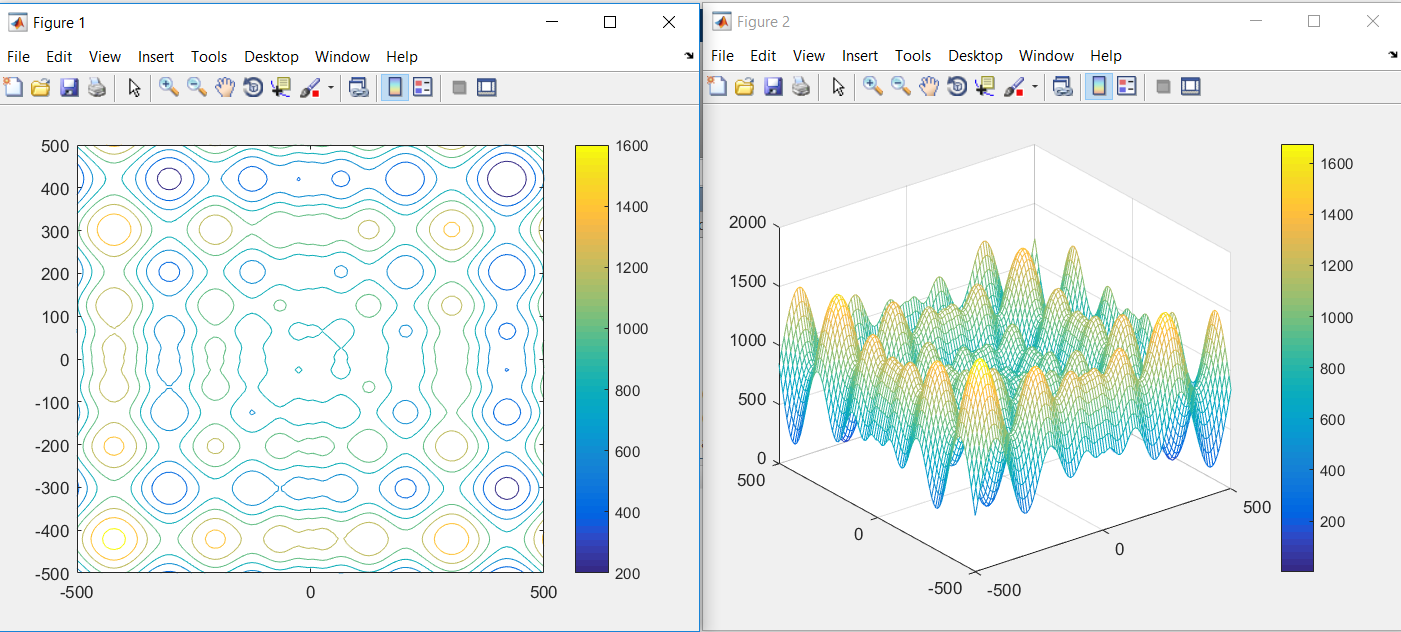
The counter and 3D plot for 500 iterations



The counter and 3D plot for 1000 iterations



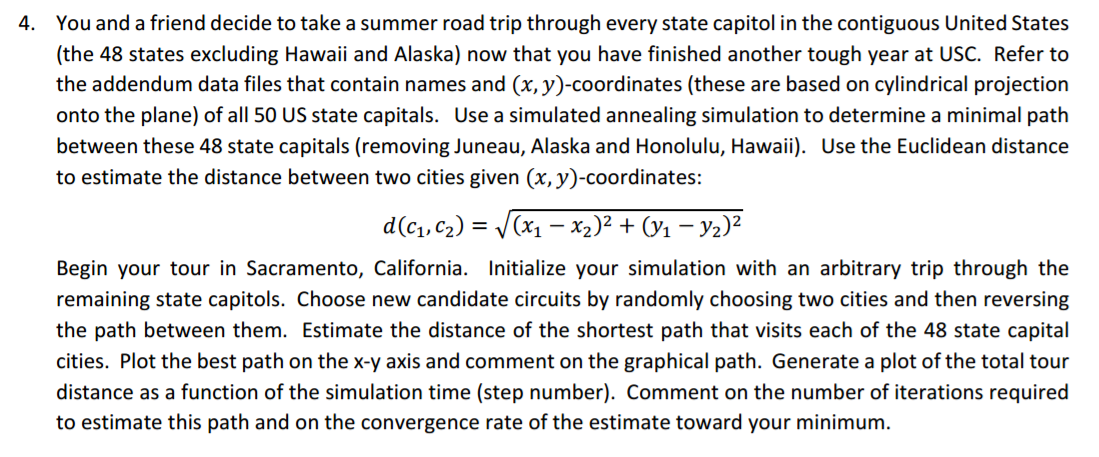
The counter and 3D plot for 10000 iterations



**Observations:**

* We can see the quality of the EM through the pdf parameters to the initial mean and sigma value we provided in the cluster.

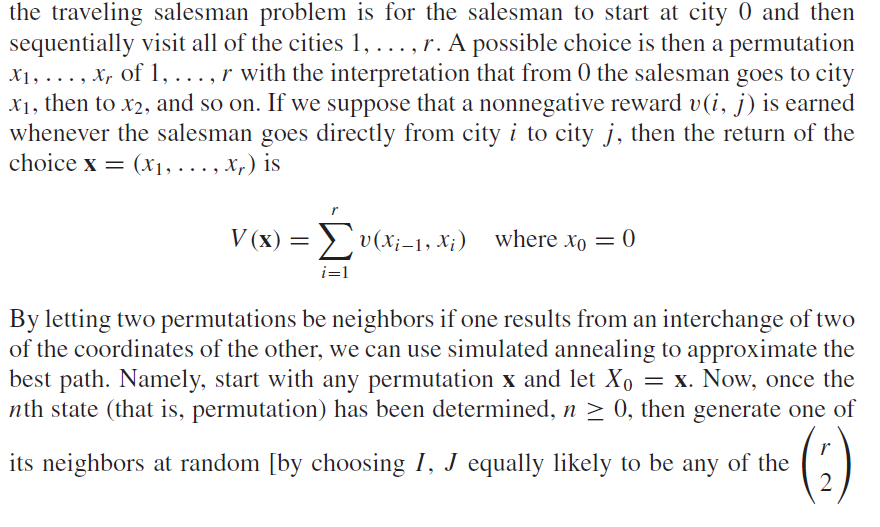
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

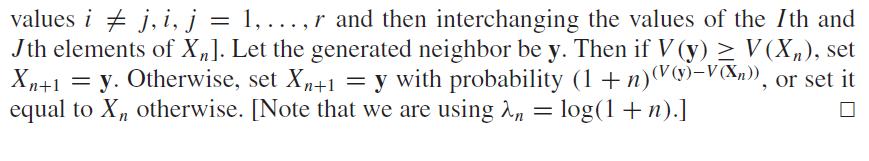


**Concept behind this problem:**

The problem uses the concept of ***Simulated annealing (explained above) and Travelling salesman problem (TSP)***.

TSP presents the task of finding the most efficient route through a set of given locations. Each location should be passed through only once, and the route should loop so that it ends at the starting location. The TSP consists of a set of "cities," or points, with the object of finding the shortest path connecting them under a given metric. The difficulty of this problem is obvious when one realizes that an n-city set has a search space of (n-1)!.





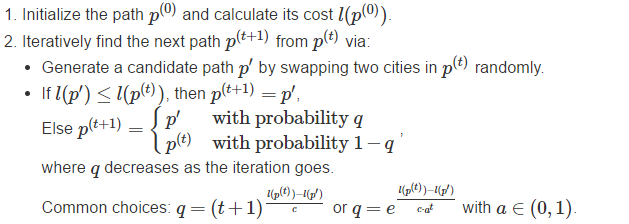
**Reference:**

**[1] Piazza**

**[2]** [**https://www.hindawi.com/journals/cin/2016/1712630/**](https://www.hindawi.com/journals/cin/2016/1712630/)

**[3] Simulation by Sheldon Ross**

**Code and Description:**



* With the above approach, we first read the given input data .txt file and then create a vector of 48x2 elements.

content = fileread('uscap\_xy.txt');

data = textscan( content, '%f %f%\*[^\n]','HeaderLines', 0) ;

x = data{1};

y = data{2};

n= 48;

city = [x y];

* We then calculate the distance from P0 to its neighbors in iterative fashion swapping whenever we meet the condition mentioned above.

distance = pdist2(city, city);

num\_iter = 10000;

c = 100;

p = [4:n 1:3];

len = 0;

for a1 = 1:n-1

len = len + distance(p(a1),p(a1+1));

end

len = len + distance(p(n),p(1));

pathHistory = zeros(num\_iter,n);

lenHistory = zeros(1,n);

% Plotting intial path

figure(1)

plot(city(:,1), city(:,2), 'ro');

xlim([min(x)-1 max(x)+1]);

ylim([min(y)-1 max(y)+1]);

hold on

line(city([p(:); p(1)],1), city([p(:); p(1)],2));

title('Initial path');

hold off

count = 0;

while(count<num\_iter)

count = count + 1;

% Create path p2 by randomly swap two cities

swap\_index = randsample(n,2);

p2 = p;

temp = p2(swap\_index(1));

p2(swap\_index(1)) = p2(swap\_index(2));

p2(swap\_index(2)) = temp;

% Cost of p2

len2 = 0;

for a1 = 1:n-1

len2 = len2 + distance(p2(a1),p2(a1+1));

end

len2 = len2 + distance(p2(n),p2(1));

q = (1+count)^((len - len2)/c);

if len2 - len <= 0

p = p2;

len = len2;

else

if rand <= q

p = p2;

len = len2;

end

end

pathHistory(count,:) = p;

lenHistory(count) = len; end

**Results and Plots:**

**Num\_of\_iterations = 1000**

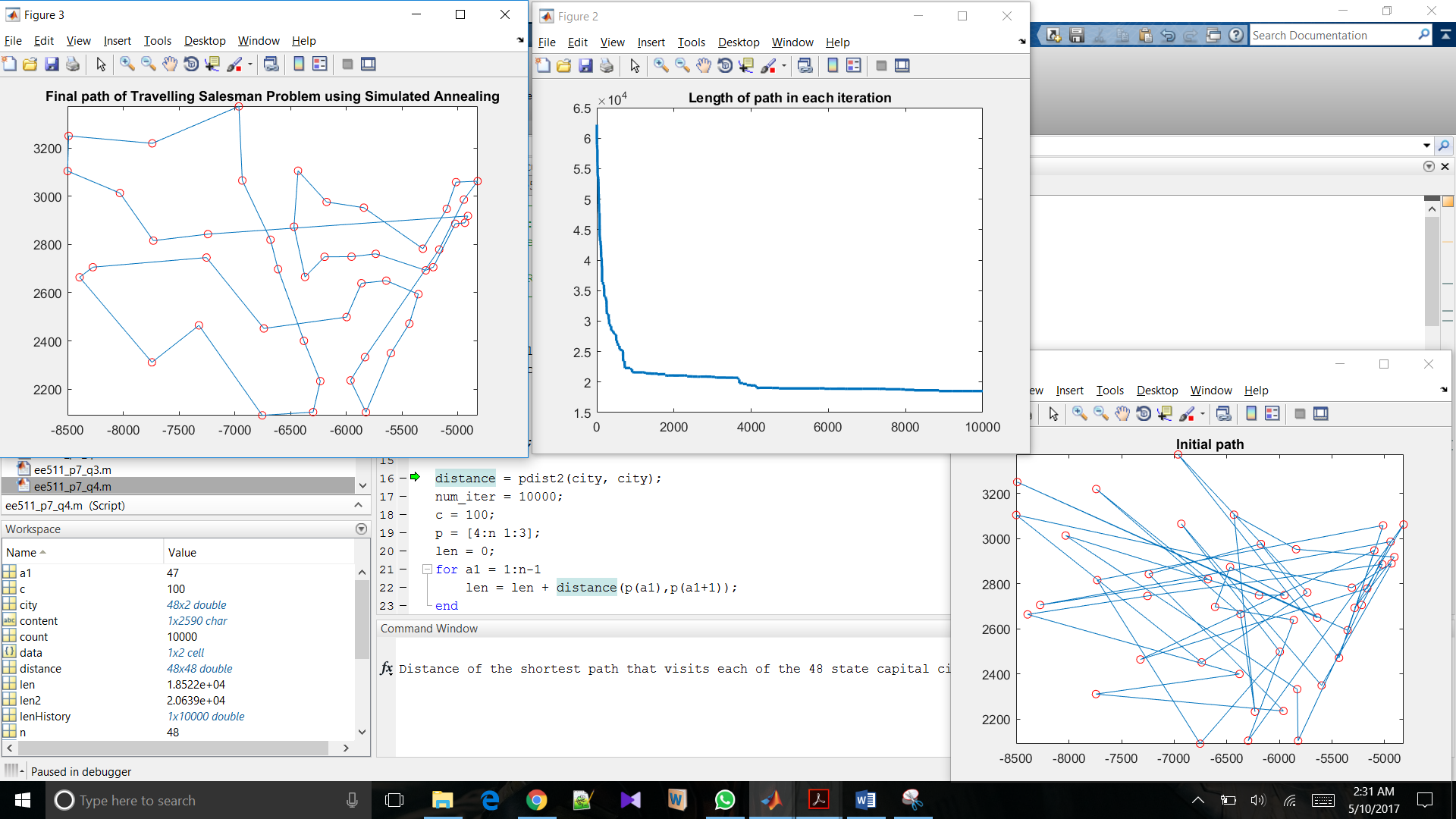
Distance of the shortest path that visits each of the 48 state capital cities is: 21673.179641>>

**Num\_of\_iterations = 10,000**

Distance of the shortest path that visits each of the 48 state capital cities is: 15999.473092K>>

**Num\_of\_iterations = 100,000**

Distance of the shortest path that visits each of the 48 state capital cities is: 16538.133280>>



**Observations:**

* We calculated final path via TSP which is the shortest path from initial to all cities.
* We also observe that since the best path(shortest) between each city is chosen, the algorithm computes the overall shortest path i.e. the quality of convergence to the shortest using this method much accurate than the other algorithms.
* As the number of iterations increases, the overall distance decreases as observed from the distance mentioned above.