```matlab
%--------READING THE DATA----------%
clear all;
clc;
PI=pi;
phiqdata=xlsread('C:\Users\Baktha\Desktop\Personal\sampleData.xlsx');
k=1;
for i=1:2:143
    PHIp(:,k)=phiqdata(:,i);
    Qp(:,k)=phiqdata(:,i+1);
    Np(:,k)=(phiqdata(:,i+1))/2;
    k=k+1;
end


%--------STATE ASSIGNMENT----------%
T=72;
S=30;
N=5;
tot=T*S; % total no of phi, q sets
z=0;
S = 30;
dim =2 ;
wind = 18;
data=phiqdata;
testdata(1:S,:) = phiqdata(1:S,:);
w(1,:) = data(1,:);
w(2,:)= data(3,:);
w(3,:) = data(5,:);
w(4,:)= data(6,:);
w(5,:) = data(8,:);
w(6,:) = data(10,:);
w(7,:)= data(11,:);
w(8,:)= data(13,:);
w(9,:) = data(15,:);
w(10,:) = data(16,:);
w(11,:) = data(18,:);
w(12,:) = data(20,:);
w(13,:) = data(21,:);
w(14,:) = data(23,:);
w(15,:) = data(25,:);
w(16,:) = data(26,:);
w(17,:) = data(28,:);
w(18,:) = data(30,:);
w_o = zeros(size(w)); % create a zero matrix of same dimentions of weight
dif = w - w_o; % initialise difference.
count = 1;                    % initialise iteration count.
while sum(sum(dif)) ~= 0 & count ~= 500
w_o = w; % remember the weights of previous iterations.
for ii = 1:S
for jj = 1:wind
eq_dist(jj) = ((testdata(ii,:)-w(jj,:)) * ((testdata(ii,:)-w(jj,:))')) ; %
equiledian distance
end
[temp,near_class(ii)] = min(eq_dist); % find the cluster which is in minimum
distance from the training exempler.
end
```

```
for ii = 1:wind
[a,B] = find(near_class == ii);
temp_sum = 0;
for jj = B
temp_sum = temp_sum + testdata(jj,:);
end
if sum(a) == 0
    count;
    ii;
end
w(ii,:) = temp_sum / sum(a);
end
dif = abs(w - w_o);
count = count+1;
%------min dist----------%
end
clear g
for i=1:wind
for j=1:S
eq_dist(j)=sqrt((w(i,:)-testdata(j,:))*((w(i,:)-testdata(j,:))'));
end
[temp,j1]=min(eq_dist);
g(i,:)=testdata(j1,:);
end


for i=1:wind
l=1;d=1;
for j=1:144
g1(i,l,d)=g(i,j);
d=d+1;
if d==3
d=1;
l=l+1;
end
end
end





wind=72;
o=18;
for k=1:o
clear g
g(:,:)=g1(k,:,:);
st=5;
for i=1:st
c(i,:)=g(i,:);
end
cl=1;
while cl==1
cl=1;
for i=1:st
```

```
for j=1:wind
dm(i,j)=sqrt((c(i,:)-g(j,:))*((c(i,:)-g(j,:))'));
end
end
sm=zeros(st,wind);
[temp,temp1]=min(dm);
for i=1:wind
sm(temp1(i),i)=1;
end
tsm=sum(sm,2);
for i=1:st
if tsm(i,1)>1
temp2=0;
for j=1:wind
if sm(i,j)==1
temp2=temp2+g(j,:);
end
end
c(i,:)=temp2/tsm(i,1);
end
end
for i=1:st
for j=1:wind
dm(i,j)=sqrt((c(i,:)-g(j,:))*((c(i,:)-g(j,:))'));
end
end
sm1=zeros(st,wind);
[temp,temp1]=min(dm);
for i=1:wind
sm1(temp1(i),i)=1;
end
if sm==sm1
cl=0;
end
end
ss(k,:)=temp1(1,:);
end

state=ss;

num=1;
for u=1:o
    for v=1:72
       statetot(num)=state(u,v);
       num=num+1;
    end
end

tot=o*wind;
%-------Initial model---------%
N=5;
for i=1:N
    pii(i)=0;
    for z=1:tot
        if(statetot(z)==i)
            pii(i)=pii(i)+1;
```

```
        end
    end
    Pi(i)=pii(i)/S;
end

for i=1:N
    for j=1:N
        AA(i,j)=0;
        for y=1:o
            for x=1:T-1
                if((state(y*x)==i)&&(state(y,(x+1))==j))
                    AA(i,j)=AA(i,j)+1;
                end
            end
        end
        A(i,j)=AA(i,j)/pii(i);
    end
end
tot=o*wind;
%finding mean
k=0;
% phitot=0;
% qtot=0;
% ntot=0;
meanofstate=zeros(5,3);
for i=1:N
phitot=0;
qtot=0;
ntot=0;
    for m=1:tot
        if(state(m)==i)
            phitot=phitot+PHIp(m);
            qtot=qtot+Qp(m);
             ntot=ntot+Np(m);
            k=k+1;
        end
    end
    phimean=phitot/k;
    qmean=qtot/k;
   % meanofstate(i,:)=[phimean,qmean];
    %ntot=5;
    nmean=ntot/k;
    meanofstate(i,:)= [phimean,qmean, nmean];
end

PQN=zeros(S,T,3);
m=1;
for i=1:S
    for j=1:T
        Qp(m)=Np(m);
        PQN(i,j,:)=[PHIp(m),Qp(m),Np(m)];
        m=m+1;
    end
end

%finding covariance
```

```
PQNtot=0;
statexy=state;
t=0;
covar=zeros(5,3,3);
for m=1:N
    for i=1:o
        for j=1:T
            if(statexy(i,j)==m)
                PQNtot=PQNtot+PQN(m,:);
                tem=[PQN(i,j,1),PQN(i,j,2),PQN(i,j,3)];
                t=t+(((tem-meanofstate(m,:))')*(tem-meanofstate(m,:)));
                k=k+1;
            end
        end
    end
    covar(m,:,:)=t/k;
end

%finding b
M=3; % phi, q, n

for i=1:S
    tem=zeros(3,3);
    for j=1:N
        for m=1:T
            t1=1/((2*3.14)^(M/2));
            for x=1:3
                for y=1:3
                    tem(x,y)=covar(j,x,y);
                end
            end
            for x=1:3
                tem2(:,x)=PQN(i,m,x);
            end
            t2=1/((det(tem))^(0.5));
            t3=tem2-meanofstate(j,:);
            t4=inv(tem);
            t5=(tem2-meanofstate(j,:))';
            B(i,j,m)= t1*t2*exp(-0.5*t3*t4*t5);
        end
    end
end


%--------------HMM calculation--------%
K = 3;
%a  = [0.2,0.2,0.15,0.15,0.1,0.1 ; 0,0.2,0.1,0.25,0.25,0.1 ;
0,0,0.15,0.15,0.2,0.2 ; 0,0,0,0.25,0.3,0.45 ; 0,0,0,0,0.62,0.38 ;
0,0,0,0,0,1];
%B  = [0.4,0.4,0.2 ; 0.25,0.45,0.3 ; 0.2,0.35,0.45 ; 0.2,0.3,0.5 ;
0.6,0.2,0.2 ; 0.1,0.4,0.5];
%Pi = [0.4,0.3,0.3,0,0,0];
sum=0;

Ob=[2,3,2,3,2,1,2,2,2,1,3,2,1,1,2,3,3,2,1];
T=length(Ob);
```

```
Beta=zeros(T,N);
Alpha=zeros(T,N);
i=0;
j=0;
t=0;
ZI=zeros(T,N,N);
nu=0.0;
Gamma=zeros(T,N);
E_T=zeros(1,N);
E_I_J=zeros(1,N);
E_Pi=zeros(1,N);
E_A=zeros(N,N);
N_E_A=zeros(N,N);
E_B=zeros(N,K);
sum1=zeros(K);
p_v=zeros(N);
status=zeros(1,N);
z=0;
n=0;
tt=0;


%Forward Algorithm
for i=1:N
    Alpha(1,i)=Pi(i) * B(i,Ob(1));
end
for t=1:T-1
    for j=1:N
        sum=0;
        for i=1:N
            sum= sum+ Alpha(t,i)*A(i,j);
        end
        Alpha(t+1,j)=sum * B(j,Ob(t+1));
    end
end
disp('The forward matrix is:');
for i=1:T
    for j=1:N
        fprintf('%.8f',Alpha(i,j));
        fprintf('    ');
    end
    fprintf('\n');
end
fprintf('\n');

%Backward Algorithm
for i=1:N
    Beta(T,i)=1;
end
for t=T-1:-1:1
    for i=1:N
        sum=0;
        for j=1:N
            sum=sum+(A(i,j)*Beta(t+1,j)*B(j,Ob(t+1)));
        end
        Beta(t,i)=sum;
```

```
    end
end
disp('The backward matrix is:');
for i=1:T
    for j=1:N
        fprintf('%.8f',Beta(i,j));
        fprintf('     ');
    end
    fprintf('\n');
end
fprintf('\n');

%Baum-Welch Algorithm
kk=0;
sum2=0;
%Calculation of ZI values
for t=1:T-1
    for i=1:N
        for j=1:N
            nu=Alpha(t,i)*B(j,Ob(t+1))*Beta(t+1,j)*A(i,j);
            sum=0;
            for z=1:N
                for n=1:N
                    sum = sum + (Alpha(t,z) *A(z,n) *B(n,Ob(t+1))
*Beta(t+1,n));
                end
            end
            ZI(t,i,j) = nu/sum;
        end
    end
end
% disp('The ZI matrix is:');
% disp(ZI);

%Gamma computation
for t=1:T
    for i=1:N
        sum=0;
        for j=1:N
            sum = sum+ZI(t,i,j);
        end
        Gamma(t,i)=sum;
    end
end
disp('The Gamma matrix is:');
for i=1:T
    for j=1:N
        fprintf('%.8f',Gamma(i,j));
        fprintf('     ');
    end
    fprintf('\n');
end
fprintf('\n');

%Expected number of transistions from state i
for i=1:N
```

```
    sum=0;
    for t=1:T-1
        sum= sum + Gamma(t,i);
    end
    E_T(i)=sum;
end
disp('Expected no of transitions from the states:');
for i=1:N
fprintf('%.4f',E_T(i));
fprintf('\n');
end

%Expected number of transitions from node i to node j
for i=1:N
    for j=1:N
        sum=0;
        for t=1:T-1
            sum= sum+ZI(t,i,j);
        end
        E_I_J(i)=sum;                       % may be a mistake (already mentioned
in the C version).............
        fprintf('Expected no of transitions from the state %d to state %d:',
i,j);
        fprintf('%.4f \n',E_T(i));
    end
end

%Computing estimated values for Pi ,A and B.
for i=1:N
    E_Pi(i)= Gamma(1,i);     % E_Pi(i)= Gamma(0,(i));
end

for i=1:N
    for j=1:N
        sum=0;
        nu=0;
        for t=1:T-1
            sum=sum+ZI(t,i,j);
            nu=nu+Gamma(t,i);
        end
        E_A(i,j) = (sum / nu)   ;
    end
end
disp('The estimated state transition matrix is:');
for i=1:N
    for j=1:N
        fprintf('%.8f',E_A(i,j));
        fprintf('     ');
    end
    fprintf('\n');
end
fprintf('\n');

%Computing the matrix B
for j=1:N      % number of states
    sum2=0;
```

```
    for kk=1:K
        sum1(kk)=0;
    end
    for t=1:T  %to traverse the observation sequence...
        for kk=1:K
            if(Ob(t) == kk)  % here one for loop will come
                sum2 = sum2+ Gamma(t,j); % overall sum ..........
                sum1(kk)= sum1(kk) + Gamma(t,j);
                break;
            end
        end
    end
    for kk=1:K
        E_B(j,kk) = (sum1(kk))/sum2;
    end
end

disp('The estimated probability matrix is:');
for i=1:N
    for j=1:K
        fprintf('%.8f',E_B(i,j));
        fprintf('     ');
    end
    fprintf('\n');
end
fprintf('\n');


%probability of visit
sum = 0;
disp(' The probability of the node being visited during the training phase');
disp(N);
for i=1:N
    if(i==1)
        p_v(i)=E_Pi(i);
    else
        sum=0;
        for j=1:(i-1)
            sum= sum + p_v(j)*(E_A(j,i)/(1-E_A(j,j)) );
        end
    end
    p_v(i)= sum + (E_Pi(i));
end

tt=1;
for i=1:N
    if(p_v(i)*100 >= 40.0)
        status(tt)=i;
        tt= tt +1;
    else
        status(i)=0;
    end
end
disp('The status during the transition is:');
disp(status);
fprintf('\n');
```

```matlab
%Normalization
sum2 = 0;
sum3 = 0;
pp = 0;
pp1 = 0;
for i=1:N
    if(i==status(pp+1)) % status(pp)
        pp=pp+1;
        for j=1:N
            N_E_A(i,j)=E_A(i,j);
        end
    else
        sum3=0;
        sum2=0;
        pp1=0;
        for j=1:N
            if(j==status(pp1+1)) %status(pp1)
                pp1=pp1+1;
                sum3= sum3 + A(i,j);
            else
                sum2= sum2 + E_A(i,j);
            end
        end
        pp1=0;
        for j=1:N
            if(j~=status(pp1+1))  %status(pp1)
                N_E_A(i,j)=(1-sum3)*(E_A(i,j)/sum2);
            else
                pp1=pp1+1;
                N_E_A(i,j)=A(i,j);
            end
        end
    end
end
disp('After Normalization:');
fprintf('\n');
disp('The estimated state transition matrix is:');
for i=1:N
    for j=1:N
        fprintf('%.8f',E_A(i,j));
        fprintf('     ');
    end
    fprintf('\n');
end
fprintf('\n');
disp('The estimated probability matrix is:');
for i=1:N
    for j=1:K
        fprintf('%.8f',E_B(i,j));
        fprintf('     ');
    end
    fprintf('\n');
end
```