Def

```
% All the declared variables and arrays have been assigned 0 unless and
% otherwise stated as in the case of arrays 'a' and 'b'. The function
% declaration has been omitted as there is no declaration part in MATLAB.
global T;global N;global K;global a;global b;global Pi;global Ob;global
Beta;global Alpha;
global i;global j;global t;global ZI;global nu;global Gamma;global E_T;global
E_I_J;global E_Pi;global N_E_A;global E_A;global E_B;global sum1;global sum;
global p_v;global m;global n;global status;global tt;
T = 19;  % T=20;
N = 6;   %N=5;
K = 3;   %k=4;
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];
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(N);
E_I_J=zeros(N);
E_Pi=zeros(N);
E_A=zeros(N,N);
N_E_A=zeros(N,N);
E_B=zeros(N,N);
sum1=zeros(K);
p_v=zeros(N);
status=zeros(N);
m=0;
n=0;
tt=0;
```

FORW

```
%A matlab function is written. The name of the function is given as 'forw'
%because the file name and the function name must be the same. Alpha is
%returned after the function body is executed.
%The function call will be of the form : Alpha = forw;

function Alpha = forw()
global N;global Pi;global Ob;global b;global T;global a;
```

```
for i=1:N
        Alpha(1,i)=Pi(i) * b(i,Ob(1)); %Alpha(0,i),  Ob(0)
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
end
```

Backw

```
%A matlab function is written. The name of the function is given as 'backw'
%because the file name and the function name must be the same. Beta is
%returned after the function body is executed.
%The function call will be of the form : Beta = backw;
function Beta = backw()
global i;global N;global T;global Beta;global t;global j;global a;global
b;global Ob;global sum;
   for i=1:N
       Beta(T,i)=1;
   end                    %Beta=ones(T,N);
   for t=T-1:-1:1         %t-T-2:1
     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
end
```

BW ALGO

```
%A matlab function is written. The name of the function is given as 'Bw_algo'
%because the file name and the function name must be the same. 'ZI , Gamma,
E_Pi , E_A' and
```

```
%'E_B' are returned after the function body is executed.

function [ZI , Gamma, E_Pi , E_A , E_B ]= Bw_algo()
global kk; global sum2; global t;global T;global ZI; global Gamma;
global E_Pi; global E_A; global E_B; global N; global m;global n;
global nu;global sum;global a;global b;global Ob;global Beta;
global E_T;global K;global sum1;global Alpha;global i;global j;
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 m=1:N
                for n=1:N
                    sum = sum + (Alpha(t,m) *a(m,n) *b(n,Ob(t+1))
*Beta(t+1,n));
                end
            end
                        ZI(t,i,j) = nu/sum;
        end
    end
end

 %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

  %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

   %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).............
           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

  %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
end
```

PV

```
%A matlab function is written. The name of the function is given as 'P_V'
%because the file name and the function name must be the same. 'status'
%matrix is returned after the function body is executed.

function [ status ]= P_V()
global sum;global N;global i;global p_v;global j;global E_Pi;global
E_A;global status;global tt;
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)n
             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;  %tt=0;
 for i=1:N
    if(p_v(i)*100 >= 40.0)
       status(tt)=i;
       tt= tt +1;
    else
        status(i)=0;
    end
 end
```

NORMAL

```
%A matlab function is written. The name of the function is given as 'noramal'
%because the file name and the function name must be the same. 'N_E_A' and
%'E_A' are returned after the function body is executed.

function [ N_E_A , E_A ]= noramal()
global sum2;global sum3;global pp;global pp1;global i;global N;global
N_E_A;global E_A;global j;global status;global a;
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
```

HMM

```
def;
global Alpha;global Beta;global ZI;global Gamma; global E_A;global
E_Pi;global E_B;global status;global N_E_A;
Alpha=forw();
Beta=backw();
[u,v,w,x,y]=Bw_algo();
ZI=u;
Gamma=v;
E_Pi=w;
E_A=x;
E_B=y;
[sta]=P_V();
status=sta;
[nonerg, erg]= noramal();
N_E_A=nonerg;
E_A=erg;
```