```
function out = AnaliseMarkovToAr(mkvStruct, eps)
% Autor: Bruno Tebaldi Q Barbosa
%
% Adaptação do codigo de Tiago Cavalcanti
% Calculate the invariant distribution of Markov chain by simulating the
% chain to reach a long-run level
%
% Imputs:
% PI: Matriz de transição do processo de markov
% z: Vetor de estados
%
%
%
% eps:
if nargin <2
   eps = 1e-8;
end


% Assume ua probabilidade igual para se iniciar em todos os estados.
prob = (1/mkvStruct.QtdStates)*ones(mkvStruct.QtdStates,1); % initial distribution of↙
states
test = 1;

% Calcula o estado final de equilibrio
while test > eps
    probst1 = mkvStruct.TransitionMatrix'*prob;
    test=max(abs(probst1-prob));
    prob = probst1;
end

% Calculate Properties of Invariant Distribution
meanm = mkvStruct.StateVector*prob;               % mean of invariant distribution
varm = ((mkvStruct.StateVector-meanm).^2)*prob;   % variance of invariant distribution

midaut1 = (mkvStruct.StateVector-meanm)'*(mkvStruct.StateVector-meanm); % cross product↙
of deviation from the
                                % mean of y_t and y_t-1

probmat = prob*ones(1,mkvStruct.QtdStates);     % each column is invariant distribution

midaut2 = mkvStruct.TransitionMatrix.*probmat.*midaut1; % product of the first two terms↙
is
                                % the joint distribution of (Y_t-1,Y_t)

autcov1 = sum(sum(midaut2));      % first-order auto-covariance

alambda = autcov1/varm;           % persistence of discrete process
asigmay = sqrt(varm);            % s.d. of discrete process
```

```
% Calculate the Asymptotic second moments of Markov chain

fprintf('______________________________________________________\n')
fprintf('                    original process     Markov chain\n')
fprintf('______________________________________________________\n')
fprintf('Persistence         %16.6f %16.6f\n', mkvStruct.AR.rho, alambda);
fprintf('Standard deviation  %16.6f %16.6f\n', mkvStruct.AR.sigma2_y^0.5, asigmay);
fprintf('______________________________________________________\n')
end
```