Stanley Ung

01143626

EE 381

**Project 6: Markov Chains**

**Problem 1:**

**Introduction:**

**Methodology:**

**Code:**

**import numpy as np**

**import matplotlib.pyplot as plt**

**N = 10\*\*4**

**n = 15**

**P = np.matrix( [[1/3, 1/3, 1/3], [1/2, 0, 1/2], [1/4, 1/4, 1/2]] )**

**D = np.matrix( [1/4, 1/2, 1/4] )**

**X = []**

**M = []**

**for j in range(0, N):**

**num = np.random.rand()**

**S = []**

**if num <= D[0, 0]:**

**state = 'R'**

**elif D[0, 0] < num <= (D[0, 0] + D[0, 1]):**

**state = 'N'**

**elif (D[0, 0] + D[0, 1]) < num:**

**state = 'S'**

**S.append(state)**

**for k in range(0, n - 1):**

**s = S[k]**

**r = np.random.rand()**

**if s == 'R':**

**if r <= P[0, 0]:**

**\_state = 'R'**

**elif P[0, 0] < r <= (P[0, 0] + P[0, 1]):**

**\_state = 'N'**

**elif (P[0, 0] + P[0, 1]) < r:**

**\_state = 'S'**

**elif s == 'N':**

**if r <= P[1, 0]:**

**\_state = 'R'**

**elif P[1, 0] < r <= (P[1, 0] + P[1, 1]):**

**\_state = 'N'**

**elif (P[1, 0] + P[1, 1]) < r:**

**\_state = 'S'**

**elif s == 'S':**

**if r <= P[2, 0]:**

**\_state = 'R'**

**elif P[2, 0] < r <= (P[2, 0] + P[2, 1]):**

**\_state = 'N'**

**elif (P[2, 0] + P[2, 1]) < r:**

**\_state = 'S'**

**S.append(\_state)**

**X.append(S)**

**for j in range(0, n):**

**x = [X[i][j] for i in range(0, N)]**

**ma = len(list(filter(lambda a : a == 'R', x)))**

**mb = len(list(filter(lambda a : a == 'N', x)))**

**mc = len(list(filter(lambda a : a == 'S', x)))**

**M.append([ma/N, mb/N, mc/N])**

**ma = [M[i][0] for i in range(0, len(M))]**

**mb = [M[i][1] for i in range(0, len(M))]**

**mc = [M[i][2] for i in range(0, len(M))]**

**nv = range(0, n)**

**plt.figure(1);**

**plt.title('Simulation results -- States R, N,S')**

**plt.xlabel('Time step (n)')**

**plt.ylabel('Prob(State)')**

**plt.legend(['State R', 'State N', 'State S'])**

**plt.plot(ma ,'\*:', mb, '\*:', mc, '\*:')**

**plt.show()**

**D = [[1/3, 1/3, 1/3], [1/2, 0, 1/2], [1/4, 1/4, 1/2]]**

**Y = [[1/4, 1/2, 1/4]]**

**for k in range(0, n - 1):**

**Y.append(np.dot(Y[k], D))**

**print(\*Y, sep='\n')**

**ya = [Y[i][0]for i in range(0, len(Y))]**

**yb = [Y[i][1]for i in range(0, len(Y))]**

**yc = [Y[i][2]for i in range(0, len(Y))]**

**plt.figure(2);**

**plt.plot(nv, ya,'o:',nv, yb,'o:',nv, yc,'o:');**

**plt.title('Results based on State Transition Matrix -- States R, N,S')**

**plt.xlabel('Time step (n)')**

**plt.ylabel('Prob(State)')**

**plt.legend(['State R', 'State N', 'State S'])**

**plt.show()**

**plt.figure(3);**

**plt.plot(nv, ma ,'\*:', nv, mb, '\*:', nv, mc, '\*:', nv, ya,'o:',nv, yb,'o:',nv, yc,'o:')**

**plt.title('Comparison: Experimental simulation & State transition matrix')**

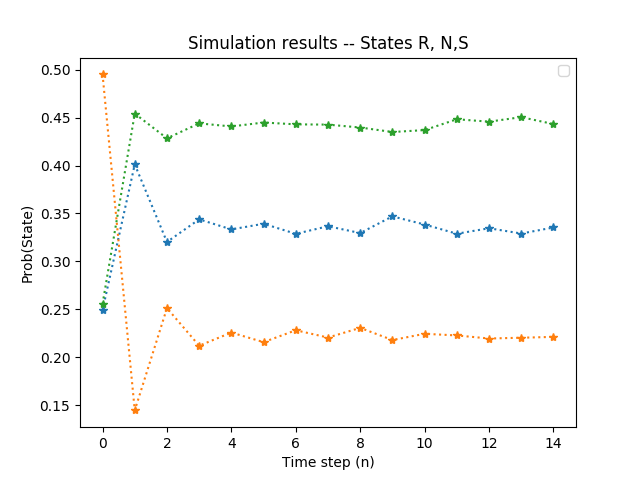
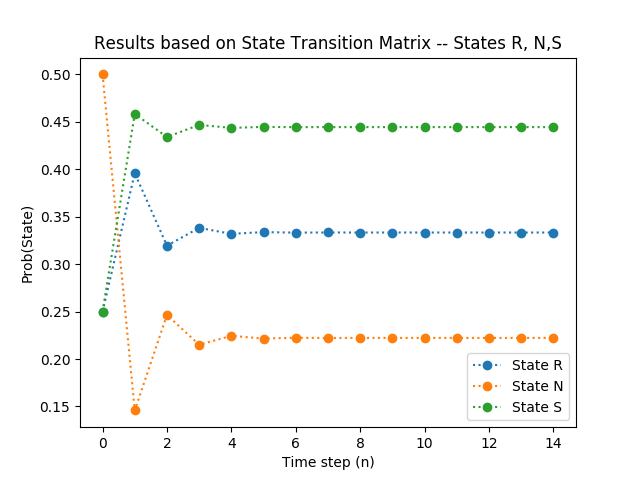
**plt.xlabel('Time step (n)')**

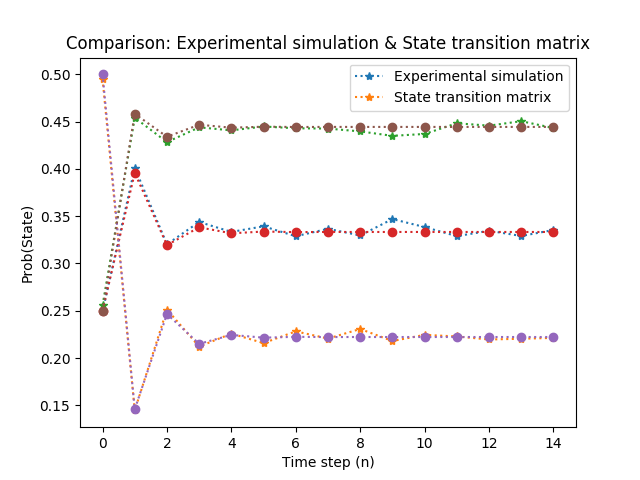
**plt.ylabel('Prob(State)');**

**plt.legend(['Experimental simulation','State transition matrix'])**

**plt.show()**

**Results and Conclusion:**

****

****

**Problem 2:**

**Introduction:**

**Methodology:**

**Code:**

**N = 10\*\*4**

**n = 20**

**P = [[0, 1, 0, 0, 0], [1/2, 0, 1/2, 0, 0], [1/3, 1/3, 0, 0, 1/3], [1, 0, 0, 0, 0], [0, 1/3, 1/3, 1/3, 0]]**

**v = [[1/5, 1/5, 1/5, 1/5, 1/5], [0, 0, 0, 0, 1]]**

**for j in range(0, 2):**

**Y = [v[j]]**

**for k in range(0, n - 1):**

**Y.append(np.dot(Y[k], P))**

**nv = range(0, n)**

**plt.figure(j);**

**plt.title(['PageRank Probabilities', ' ; s0 = [', str(v[0]), ']']);**

**plt.xlabel('Time step (n)');**

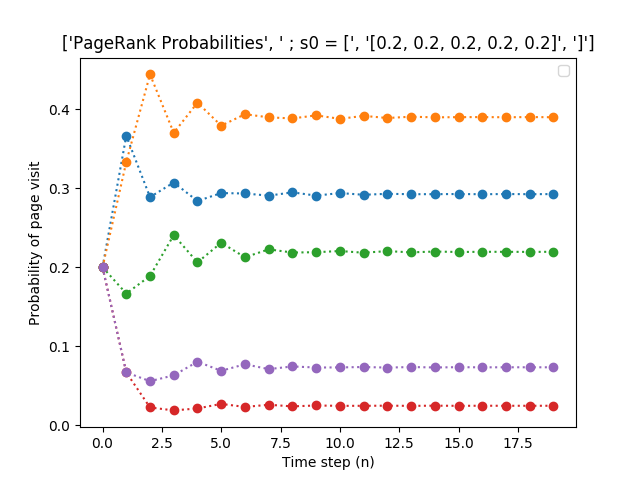
**plt.ylabel('Probability of page visit')**

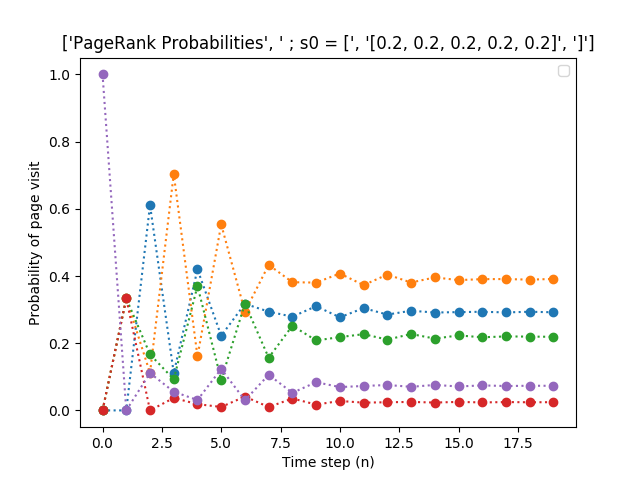
**plt.legend(['A', 'B', 'C', 'D', 'E'])**

**plt.plot(nv, Y,'o:')**

**plt.show()**

**Results and Conclusion:**

****

****