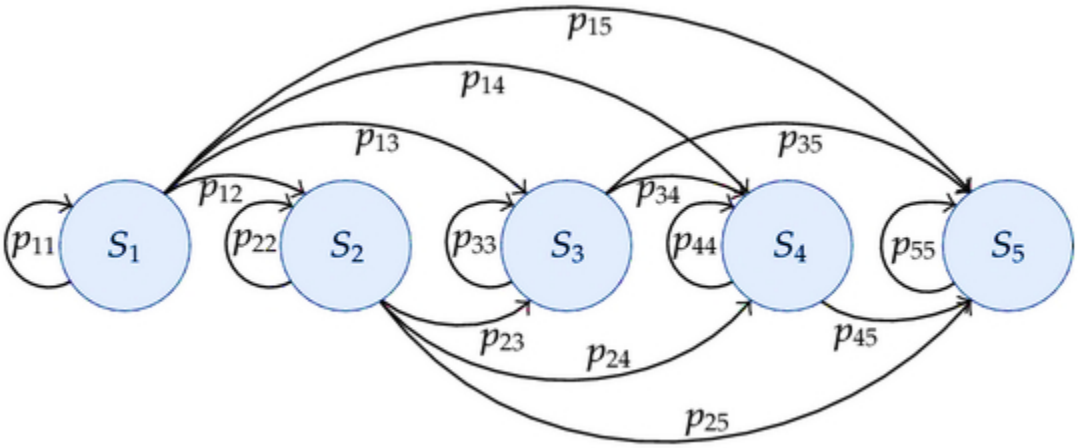
IE 203 OPERATIONS RESEARCH II

PROJECT REPORT

Calculating Steady-State Distributions with Two Methods



prepared by

MELİS TUNCER

2019402156

Instructor: Z. Caner Taşkın

TABLE OF CONTENTS

Aim of The Study

The aim of this study is to calculate steady-state distributions of 3 transition probability matrices belonging to 3 ergodic Markov Chains with two methods, to compare and interpret the results.

Initialization

As the starting point, a transition probability matrix (P) of an ergodic MC should be obtained.

$$\mathbf{P} = \begin{array}{c} \\ 0 \\ 1 \\ \vdots \\ n \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & \dots & n \end{array} \\ \begin{pmatrix} p_{00} & p_{01} & \dots & p_{0n} \\ p_{10} & p_{11} & \dots & p_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ p_{n0} & p_{n1} & \dots & p_{nn} \end{pmatrix} \end{array}$$

The transition probability matrix must be in compliance with the facts that;

- The MC is aperiodic.
- The MC is irreducible.

In order to ensure these statements, the probability matrix is created such that;

- $pij > 0$ (1) (it is possible to go from every state to every state)
- $\Sigma_j\, pij = 1\ for\ \forall i.$ (2)

In this study, we are asked to create 3 transition probability matrices having dimensions 5x5 (P1), 25x25 (P2), and 50x50 (P3). The algorithm that I used takes an input "n" and forms a matrix with n rows and n columns. Then for each row i of the matrix, n many arbitrary numbers (indicating pij) between 0 and 1 are created. This step guarantees (1). In order to ensure that the sum of each row adds up to 1, the pij values of each row are manipulated in the following way: pij = pij / $\Sigma_j\, pij$. With this, (2) is also guaranteed. The R-code of the algorithm can be found in Appendix A.

Method I: Monte Carlo Simulation

Now that we have created 3 different transition probability matrices by the previous algorithm. One possible way of obtaining steady-state distributions of these MC's is to simulate a process with a very large number of steps. In this study, the step size is given as $M = 2x10^5$. To obtain steady-state distributions, we need to count the number of times that we observe state i in the sequence (M_i) and divide it by M. Here is my algorithm to determine steps and steady-state distributions:

- Given an nxn transition probability matrix (P), [states] vector holds all the states from 1 to n. For instance, [states] = [1, 2, 3, 4, 5] for P1.

- [X] vector keeps track of all the states that we visit. For instance, the second element of X being 4 means that the algorithm chose to go to the 4^{th} state in the second step.

- CumProbs matrix is obtained by converting the rows of (P) into cumulative probabilities. For instance; if row i of P is given by [0.2, 0.1, 0.5, 0.2], row i of CumProbs will be: [0.2, 0.3, 0.8, 1].

- The starting state is chosen by sampling 1 number from [states] vector, and is added to [X] to be its first element.

- XC is the current state that we are. In order to determine the next state, a random number r between 0 and 1 is created. Then, the cumulative probabilities row of our current state is iterated from its first element to the last element. The index of the first value than which r is smaller is returned as the next step, called Xnext. For instance, let the algorithm be in the 3^{rd} state currently, and the 3^{rd} row of CumProbs be as follows: [0.2, 0.3, 0.8, 1]. If the random number r is turned out to be 0.273, the next state will be 2 which is the index of 0.3, the first element of the row that is bigger than r.

- After Xnext is determined; the loop breaks, Xnext is added to [X], and XC becomes Xnext. The same process continues until M many steps of the simulation is placed into [X]. In the end, the number of times that we observe state i (the number of times that "i" is seen among the elements of [X]) is divided by M, forming the [pilong] vector. This vector holds the steady-state distributions of each state i at its i^{th} index.

The R-code of the algorithm is available in Appendix A.

Method II: Matrix Multiplication

Another efficient way to calculate steady-state distribution is to multiply the transition probability matrix adequately many times. The following procedure is used to obtain these values:

- MC is the current matrix that we have, which is P at the beginning. At each iteration, MC is multiplied with itself and the resulting matrix becomes the new MC. This process continues until the difference between the column means of MC and an arbitrary row of MC is small enough. The limit of acceptable difference is called error tolerance (e), whose value is given by 5×10^{-4}.

- When the multiplication process stops, the column means of the last obtained matrix is returned. The returned vector again holds the steady-state distributions of each state i at its i^{th} index.

Further information regarding the R-code can be found in Appendix A.

Comparing the Results of the Two Methods

The resulting steady-state distributions of 3 markov chains calculated with 2 methods can be seen in Appendix B. It can also be seen that the results are very close to each other.

To be more precise in comparison, I have applied the paired t-test, that is commonly used in statistics to determine if the difference between the averages of 2 groups of data is statistically significant or not. Hereafter, I will refer the resulting steady-state distributions of Monte Carlo Simulation method as **data 1** and the resulting steady-state distributions of Matrix Multiplication method as **data 2**.

The tests are constructed such that;

Hypothesized claim: **The average of the componentwise difference between the elements of data 1 and data 2 is zero.**

For matrix P1 the results at 0.05 level of significance are as follows:

- The average of the differences between data 1 and data 2 is 4e-8.
- The standard deviation of the differences between data 1 and data 2 is 9.14e-4.
- The p-value is 0.9999, which is a very strong evidence in favor of the hypothesis.
- The difference between the averages of data 1 and data 2 **is not big enough to be statistically significant**.

For matrix P2 the results at 0.05 level of significance are as follows:

- The average of the differences between data 1 and data 2 is -1.04e-8
- The standard deviation of the differences between data 1 and data 2 is 4.99e-4.
- The p-value is 0.9999, which is a very strong evidence in favor of the hypothesis.
- The difference between the averages of data 1 and data 2 **is not big enough to be statistically significant**.

For matrix P3 the results at 0.05 level of significance are as follows:

- The average of the differences between data 1 and data 2 is -1.43e-5.
- The standard deviation of the differences between data 1 and data 2 is 2.36e-4.
- The p-value is 0.6702, which is a very strong evidence in favor of the hypothesis.
- The difference between the averages of data 1 and data 2 **is not big enough to be statistically significant**.

As can also be seen from the t-test results, the two methods brought very close results. Statistically speaking, for each of the 3 markov chains, we are 95% confident that the two methods calculate the same steady-state distribution values.

Although the two methods revealed approximately same results, one should take into account another aspect of the algorithms, which is the calculation time. The first thing that I noticed when I ran the code was the difference between the execution times of the 2 methods.

The first method, Monte Carlo Simulation is working extremely slower than Matrix Multiplication. For the easiest case where only 5 states exist (P1), the execution of Monte Carlo Simulation with M = 2e5 took almost 1.5 minutes, while the execution of Matrix Multiplication is taking less than a second for all matrices P1, P2, and P3. Noticing this, I ran the code again by reducing the step size. This time, Monte Carlo Simulation took 21 seconds with M = 1e5, and 6 seconds with M = 5e4. The results are still not as good as the Matrix Multiplication regarding the calculation time. Even more, we are getting away from the true results as we decrease the step size. Since the sample size should be very large for a simulation to get relatively accurate results, decreasing the step size to obtain a smaller execution time is not convenient.

Even though the results of the two methods comply with each other, I conclude that the Matrix Multiplication method is more efficient than the Monte Carlo Simulation with regards to the execution time.

Converting One of the States into an Absorbing State

Having created 3 transition probability matrices P1, P2, and P3; converting one state to an absorbing state is done by the following procedure:

- One state, say x, is chosen by sampling 1 number from the sequence of states : 1, 2, … , n.

- The x^{th} row of the input matrix is filled with zeros.

- The [x, x] th entry of the matrix is set to one.

The R-code of the algorithm is available in Appendix A.

- Resulting Matrices

After the code is run,

- ✓ state 3 is chosen arbitrarily to be absorbing for P1,
- ✓ state 10 is chosen arbitrarily to be absorbing for P2,
- ✓ state 6 is chosen arbitrarily to be absorbing for P3.

Comments on the Results Of Two Methods with Absorbing States

The resulting steady-state distributions of 3 markov chains, with one of the states is absorbing, calculated with 2 methods can be seen in Appendix B.

Absorbing State Probabilities

- As expected, all the steady-state distributions are zero everywhere except the absorbing state, where it is equal to 1.

- To put in another way, the long-term probability that the system will be each state is independent from the starting state, when the markov chain includes only one absorbing state.

- That is because when the system enters the absorbing state, it cannot leave. In addition, in our example, the initial markov chain was ergodic, which means all states were communicating with each other. Therefore, it was possible to go to the absorbing state from each states. Since the system will inevitably enter the absorbing state at a point, the steady-state probability of the absorbing state will be equal to 1 independent from the starting state, and all the other probabilities will be zero.

Comparing Results

- The two methods again printed out the same results with each other. The steady-state probability of,
    - State 3 is 1 for P1, since it is the absorbing state of P1,
    - State 10 is 1 for P2, since it is the absorbing state of P2,
    - State 6 is 1 for P3, since it is the absorbing state of P3,

    and all the other probabilities are zero in the results of both methods.

- However, time complexity again differs as stated in the previous comparison. The Monte Carlo Simulation method is still not efficient regarding the calculation time.

APPENDIX A

```
GenTransMatrix <- function(n){                  # generates a transition
  matr <- matrix(nrow=n, ncol=n)                probability matrix of size n
  for (i in 1:n){
    rowi <- runif(n)
    matr[i,] <- rowi/sum(rowi)}
matr
}
```

```
makeabsrb <- function(matrix){                  # converts one of the
  n <- dim(matrix)[1]                           states of the input matrix
  x <- sample(1:n,1)                            into an absorbing state
  matrix[x,] <- 0
  matrix[x,x] <- 1
  matrix
}
```

```
magnitude <- function(vec){                     # calculates the magnitude
  sum <- 0                                      of input vector
  for (i in 1:length(vec)){
    sum <- sum + (vec[i])^2}
  sqrt(sum)
}
```

Monte Carlo Simulation

```
simulMonteCarlo <- function(matrix, M){
  n <- dim(matrix)[1]                    # n is the size of matrix
  states <- vector()
  for (i in 1:n){states[i] = i}          # possible states vector

  CumProbs <- matrix(nrow=n, ncol=n)
  for (i in 1:n){CumProbs[i,] <- cumsum(matrix[i,])}
  # cumulative probabilities of each state

  X <- vector()              # monte carlo sequence that we create
  X1 <- c(sample(states, 1))
  # starting state is chosen arbitrarily
  X <- c(X, X1)     # starting state is added to the sequence
  XC <- X1          # XC is the current state

  for (i in 1:(M-1)){        # for loop to determine remaining M-1
  r <- runif(1)                states using cumulative probabilities
  for (j in 1:n){
    if (r<CumProbs[XC,j]){
      Xnext <- j
      break}}
  X <- c(X, Xnext)
  XC <- Xnext
  }
  pilong <- vector()      # vector holding steady-state probabilities
  for(i in 1:n){pilong[i] <- mean(X==i)}
  return(pilong)
  }
```

Matrix Multiplication

```
MatrMulti <- function(matrix, e){
  MC <- matrix
  n <- dim(matrix)[1]
  r <- sample(1:n, 1)
  while (magnitude(colMeans(MC)-MC[r,])>e){    # stopping condition
    MC <- MC %*% MC
    r <- sample(1:n, 1)
  }
  colMeans(MC)
}
```

Running Codes

```
P1 <- GenTransMatrix(5)
P2 <- GenTransMatrix(25)
P3 <- GenTransMatrix(50)
```

matrices of size 5, 25, and 50 are created

```
MonteP1 <- simulMonteCarlo(P1, 200000)
MonteP2 <- simulMonteCarlo(P2, 200000)
MonteP3 <- simulMonteCarlo(P3, 200000)
```

monte carlo simulation is conducted for each matrix with step size = 200000

```
MultiP1 <- MatrMulti(P1, 0.0005)
MultiP2 <- MatrMulti(P2, 0.0005)
MultiP3 <- MatrMulti(P3, 0.0005)
```

matrix multiplication is done for each matrix with error size = 0.0005

```
P1a <- makeabsrb(P1)
P2a <- makeabsrb(P2)
P3a <- makeabsrb(P3)

MonteP1a <- simulMonteCarlo(P1a,
200000)
MonteP2a <- simulMonteCarlo(P2a, 200000)
MonteP3a <- simulMonteCarlo(P3a, 200000)

MultiP1a <- MatrMulti(P1a, 0.0005)
MultiP2a <- MatrMulti(P2a, 0.0005)
MultiP3a <- MatrMulti(P3a, 0.0005)
```

one state of each matrix is made absorbing, calculations by two methods repeated with new matrices

APPENDIX B

P1 Matrix Created

	V1	V2	V3	V4	V5
1	0.3822952092	0.1637163	0.2179808	0.01510197	0.2209057
2	0.1596607196	0.1974600	0.2048309	0.20379583	0.2342525
3	0.2844236025	0.2064654	0.1907297	0.03490454	0.2834768
4	0.0007811996	0.3396620	0.3805075	0.11754884	0.1615005
5	0.2844329454	0.2157224	0.2144846	0.11175859	0.1736014

P1a Matrix Created (P1 With One Absorbing State)

	V1	V2	V3	V4	V5
1	0.3822952092	0.1637163	0.2179808	0.01510197	0.2209057
2	0.1596607196	0.1974600	0.2048309	0.20379583	0.2342525
3	0.0000000000	0.0000000	1.0000000	0.00000000	0.0000000
4	0.0007811996	0.3396620	0.3805075	0.11754884	0.1615005
5	0.2844329454	0.2157224	0.2144846	0.11175859	0.1736014

P2 Matrix Created (Columns 1-13)

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
1	0.012212049	0.0306710538	0.032774138	0.0656209131	0.092016929	0.039152751	0.003001489	0.0185641632	0.024136256	0.069376426	0.070686708	0.0194306163	0.0935097133
2	0.052788570	0.0424306092	0.040776982	0.0726676809	0.040093141	0.004298542	0.055949159	0.0517835361	0.022746245	0.039953823	0.045692973	0.0066138747	0.0287407098
3	0.030304933	0.0687330883	0.076390319	0.0326233521	0.017789371	0.017702303	0.011714687	0.0641452504	0.048390340	0.060921037	0.046257658	0.0024181687	0.0527337584
4	0.036031822	0.0238205333	0.074198330	0.0184201434	0.033273123	0.059528226	0.056709396	0.0327054335	0.021967585	0.037022387	0.068306193	0.0042140674	0.0687623645
5	0.064101371	0.0098172777	0.041525412	0.0633204285	0.017941578	0.026440425	0.057427152	0.0506476369	0.059619337	0.035118134	0.005046832	0.0560210934	0.0102122493
6	0.023771390	0.0508811349	0.036272881	0.0786802237	0.037412421	0.050836806	0.082382841	0.0633519931	0.031377575	0.082660784	0.001373866	0.0195791826	0.0672349149
7	0.031780981	0.0037928992	0.034353979	0.0800778826	0.036219015	0.070782914	0.028050768	0.0364557378	0.019342728	0.002693438	0.039978742	0.0377008025	0.0384671287
8	0.063767371	0.0586719112	0.056925468	0.0357472004	0.078460657	0.023540558	0.026370438	0.0065289357	0.054619518	0.026553426	0.030998621	0.0519587799	0.0474495465
9	0.008096438	0.0424615868	0.027361371	0.0765599691	0.074990647	0.055211104	0.045039344	0.0517650682	0.041941087	0.021980035	0.033278471	0.0013192569	0.0407059222
10	0.010882263	0.0036000839	0.045575207	0.0766934697	0.049056216	0.041506431	0.037404088	0.0083475029	0.050852648	0.020303505	0.043232738	0.0796862952	0.0084110087
11	0.032863304	0.0622051577	0.023608691	0.0475093771	0.013959248	0.021213382	0.063697516	0.0145396975	0.046962145	0.001067032	0.085319565	0.0334838337	0.0682442655
12	0.045150591	0.0140274206	0.055022851	0.0445229457	0.008290718	0.044410753	0.061548517	0.0108474953	0.057582863	0.061884763	0.046944493	0.0397123939	0.0425783564
13	0.013280368	0.0553739327	0.052562367	0.0166994767	0.039686767	0.027827138	0.030240209	0.0051272400	0.055874595	0.052511718	0.061189376	0.0453694171	0.0565431062
14	0.002597212	0.0659495667	0.059940137	0.0626582562	0.037829960	0.052014366	0.069970058	0.0208675270	0.043833035	0.065262766	0.059994458	0.0312394063	0.0661996966
15	0.022441812	0.0342560066	0.055563512	0.0228478852	0.015758449	0.017046803	0.062160321	0.0686465009	0.062747062	0.058686979	0.038241219	0.0445417146	0.0214893532
16	0.012904971	0.0235700725	0.007199595	0.0791232265	0.002202549	0.058341916	0.020405403	0.0749507694	0.014678753	0.033014697	0.047608484	0.0437542796	0.0406020474
17	0.013182670	0.0078666370	0.054264450	0.0007536338	0.027547838	0.029991286	0.043021307	0.0545650580	0.078493289	0.061360573	0.055828509	0.0946288239	0.0248895019
18	0.004517996	0.0595748529	0.056822588	0.0246367301	0.033210871	0.057313285	0.053436886	0.0452956556	0.020481419	0.028203250	0.028677932	0.0160718487	0.0227538507
19	0.001968352	0.0421410212	0.036790558	0.0390956512	0.003787270	0.028889654	0.069928818	0.0691559242	0.052006357	0.029568528	0.007025542	0.0061469199	0.0429415258
20	0.006653066	0.0464292327	0.006220666	0.0047442115	0.048974862	0.037125878	0.024447104	0.0411044316	0.037040267	0.035970442	0.052874867	0.0045066684	0.0328807202
21	0.069303211	0.0348879208	0.001598769	0.0041995718	0.067393808	0.045217608	0.006055220	0.0432809256	0.024866714	0.042772381	0.016638797	0.0156445535	0.0606276643
22	0.012819689	0.0004384623	0.011337347	0.0298622254	0.077119298	0.058584542	0.085256075	0.0009901843	0.003845788	0.043999272	0.036902148	0.0636414665	0.0621987185
23	0.054780809	0.0348223390	0.020486777	0.0630092939	0.063952046	0.063837423	0.063605675	0.0197431116	0.023755256	0.013056045	0.027417436	0.0296353068	0.0652327332
24	0.045355110	0.0135741112	0.036498710	0.0372144381	0.059743245	0.009871842	0.072230948	0.0199211531	0.058031245	0.038445571	0.070763059	0.0685675710	0.0001808498
25	0.065158737	0.0484585792	0.050995710	0.0072802711	0.057903020	0.003423997	0.074813208	0.0015955589	0.022853828	0.029349829	0.063674687	0.0009080479	0.0667822798

P2 Matrix Created (Columns 14-25)

V14	V15	V16	V17	V18	V19	V20	V21	V22	V23	V24	V25
0.039071354	0.061965637	0.017567251	0.022843437	0.05586194	0.0131803043	0.0008406411	0.018388617	0.0801698114	0.0102998857	0.051395361	0.057262552
0.036241042	0.068671876	0.046052872	0.023286503	0.02653367	0.0594907171	0.0314025615	0.024100432	0.0845395745	0.0763154026	0.011397804	0.007431698
0.036343610	0.015634060	0.058254671	0.068940020	0.03502252	0.0530351692	0.0538208120	0.034786464	0.0717706728	0.0065111489	0.015593101	0.020163484
0.064903229	0.073992678	0.048578745	0.039079231	0.03387034	0.0230652142	0.0123126826	0.007308715	0.0595522642	0.0184620897	0.010051536	0.073863669
0.039437891	0.006641166	0.003169252	0.092887474	0.05937934	0.0001944352	0.0211960882	0.070353779	0.0864413492	0.0487282660	0.073305014	0.001027014
0.020263486	0.004424652	0.012272275	0.001202473	0.01797174	0.0443074114	0.0368924500	0.026112631	0.0423034839	0.0573432963	0.021465728	0.089624364
0.038830488	0.075241580	0.063517388	0.025993456	0.07980768	0.0679881158	0.0337167307	0.030327906	0.0033668201	0.0154511801	0.060463716	0.045597923
0.072349223	0.008570857	0.021845412	0.079078120	0.01644535	0.0586633036	0.0066868954	0.064423497	0.0429234088	0.0268640088	0.011930653	0.028626842
0.014598340	0.030623183	0.059030829	0.001586694	0.06238343	0.0103549218	0.0640858798	0.065211921	0.0155359679	0.0448349816	0.035463825	0.075579723
0.073208286	0.067209225	0.004018789	0.031950120	0.07035872	0.0591480503	0.0480723590	0.047677759	0.0001216458	0.0240576052	0.033294262	0.065331722
0.029595055	0.012002106	0.040809434	0.069327102	0.05229019	0.0196233188	0.0577955370	0.071963952	0.0113563097	0.0210999419	0.052796175	0.046667666
0.063037903	0.029224594	0.047903288	0.064325651	0.03818162	0.0402062962	0.0055936045	0.024793344	0.0430598216	0.0066082310	0.045060730	0.059480757
0.049069657	0.015221212	0.019039900	0.057582098	0.01979131	0.0579873485	0.0457339681	0.029284249	0.0539804924	0.0517585128	0.051466410	0.036799134
0.008719531	0.018483837	0.024942460	0.070449426	0.02839155	0.0246714340	0.0200738033	0.045799920	0.0117695705	0.0502605567	0.005551154	0.052530316
0.023506122	0.029071040	0.051810597	0.063656905	0.02856555	0.0213747145	0.0522716312	0.012239827	0.0669458502	0.0569015822	0.007537236	0.061691323
0.017863894	0.026841689	0.064164393	0.062484608	0.04038548	0.0512802208	0.0276889313	0.008759161	0.0666350062	0.0759642631	0.083259492	0.016316094
0.037237279	0.061907234	0.082779099	0.053322567	0.03718934	0.0382441179	0.0053855382	0.011196424	0.0571629689	0.0301907966	0.037032034	0.001959026
0.037444900	0.004888026	0.065567044	0.058508846	0.06800701	0.0694226480	0.0118324701	0.054042435	0.0525509670	0.0280339784	0.057733865	0.040970646
0.072510929	0.014370422	0.077046081	0.026323828	0.08076053	0.0519625431	0.0730240961	0.045118646	0.0637789141	0.0008786291	0.048365631	0.016413631
0.065464404	0.047608792	0.067851034	0.028336367	0.05845722	0.0746728748	0.0629365575	0.039238627	0.0164006086	0.0399394473	0.042101375	0.078020271
0.046461551	0.063601667	0.051102912	0.076618048	0.06161793	0.0274405206	0.0049686379	0.077980290	0.0339216509	0.0249267895	0.057927130	0.040945733
0.005062167	0.047079058	0.035574837	0.048269039	0.07670249	0.0388689358	0.0318847061	0.025241010	0.0394562083	0.0509735390	0.028692982	0.085199813
0.027740128	0.037276867	0.072635984	0.050047847	0.02256215	0.0352868810	0.0481248120	0.052938686	0.0337635392	0.0150607025	0.041276979	0.019951175
0.039284589	0.066428418	0.019854764	0.075223191	0.02375995	0.0460792552	0.0223858828	0.013470713	0.0162487574	0.0265632962	0.080186735	0.040116599
0.075402066	0.007319377	0.035978055	0.071443356	0.02810298	0.0662837391	0.0128719037	0.017004229	0.0703390559	0.0523438882	0.035661860	0.034051741

P2a Matrix Created (P2 With One Absorbing State) (Columns 1-13)

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
1	0.012212049	0.0306710538	0.032774138	0.0656209131	0.092016929	0.039152751	0.003001489	0.0185641632	0.024136256	0.069376426	0.070686708	0.0194306163	0.0935097133
2	0.052788570	0.0424306092	0.040776982	0.0726676809	0.040093141	0.004298542	0.055949159	0.0517835361	0.022746245	0.039953823	0.045692973	0.0066138747	0.0287407098
3	0.030304933	0.0687330883	0.076390319	0.0326233521	0.017789371	0.017702303	0.011714687	0.0641452504	0.048390340	0.060921037	0.046257658	0.0024181687	0.0527337584
4	0.036031822	0.0238205333	0.074198330	0.0184201434	0.033273123	0.059528226	0.056709396	0.0327054335	0.021967585	0.037022387	0.068306193	0.0042140674	0.0687623645
5	0.064101371	0.0098172777	0.041525412	0.0633204285	0.017941578	0.026440425	0.057427152	0.0506476369	0.059619337	0.035118134	0.005046832	0.0560210934	0.0102122493
6	0.023771390	0.0508811349	0.036272881	0.0786802237	0.037412421	0.050836806	0.082382841	0.0633519931	0.031377575	0.082660784	0.001373866	0.0195791826	0.0672349149
7	0.031780981	0.0037928992	0.034353979	0.0800778826	0.036219015	0.070782914	0.028050768	0.0364557378	0.019342728	0.002693438	0.039978742	0.0377008025	0.0384671287
8	0.063767371	0.0586719112	0.056925468	0.0357472004	0.078460657	0.023540558	0.026370438	0.0065289357	0.054619518	0.026553426	0.030998621	0.0519587799	0.0474495465
9	0.008096438	0.0424615868	0.027361371	0.0765599691	0.074990647	0.055211104	0.045039344	0.0517650682	0.041941087	0.021980035	0.033278471	0.0013192569	0.0407059222
10	0.000000000	0.0000000000	0.000000000	0.0000000000	0.000000000	0.000000000	0.000000000	0.0000000000	0.000000000	1.000000000	0.000000000	0.0000000000	0.0000000000
11	0.032863304	0.0622051577	0.023608691	0.0475093771	0.013959248	0.021213382	0.063697516	0.0145396975	0.046962145	0.001067032	0.085319565	0.0334838337	0.0682442655
12	0.045150591	0.0140274206	0.055022851	0.0445229457	0.008290718	0.044410753	0.061548517	0.0108474953	0.057582863	0.061884763	0.046944493	0.0397123939	0.0425783564
13	0.013280368	0.0553739327	0.052562367	0.0166994767	0.039686767	0.027827138	0.030240209	0.0051272400	0.055874595	0.052511718	0.061189376	0.0453694171	0.0565431062
14	0.002597212	0.0659495667	0.059940137	0.0626582562	0.037829960	0.052014366	0.069970058	0.0208675270	0.043833035	0.065262766	0.059994458	0.0312394063	0.0661996966
15	0.022441812	0.0342560066	0.055563512	0.0228478852	0.015758449	0.017046803	0.062160321	0.0686465009	0.062747062	0.058686979	0.038241219	0.0445417146	0.0214893532
16	0.012904971	0.0235700725	0.007199595	0.0791232265	0.002202549	0.058341916	0.020405403	0.0749507694	0.014678753	0.033014697	0.047608484	0.0437542796	0.0406020474
17	0.013182670	0.0078666370	0.054264450	0.0007536338	0.027547838	0.029991286	0.043021307	0.0545650580	0.078493289	0.061360573	0.055828509	0.0946288239	0.0248895019
18	0.004517996	0.0595748529	0.056822588	0.0246367301	0.033210871	0.057313285	0.053436886	0.0452956556	0.020481419	0.028203250	0.028677932	0.0160718487	0.0227538507
19	0.001968352	0.0421410212	0.036790558	0.0390956512	0.003787270	0.028889654	0.069928818	0.0691559242	0.052006357	0.029568528	0.007025542	0.0061469199	0.0429415258
20	0.006653066	0.0464292327	0.006220666	0.0047442115	0.048974862	0.037125878	0.024447104	0.0411044316	0.037040267	0.035970442	0.052874867	0.0045066684	0.0328807202
21	0.069303211	0.0348879208	0.001598769	0.0041995718	0.067393808	0.045217608	0.006055220	0.0432809256	0.024866714	0.042772381	0.016638797	0.0156445535	0.0606276643
22	0.012819689	0.0004384623	0.011337347	0.0298622254	0.077119298	0.058584542	0.085256075	0.0009901843	0.003845788	0.043999272	0.036902148	0.0636414665	0.0621987185
23	0.054780809	0.0348223390	0.020486777	0.0630092939	0.063952046	0.063837423	0.063605675	0.0197431116	0.023755256	0.013056045	0.027417436	0.0296353068	0.0652327332
24	0.045355110	0.0135741112	0.036498710	0.0372144381	0.059743245	0.009871842	0.072230948	0.0199211531	0.058031245	0.038445571	0.070763059	0.0685675710	0.0001808498
25	0.065158737	0.0484585792	0.050995710	0.0072802711	0.057903020	0.003423997	0.074813208	0.0015955589	0.022853828	0.029349829	0.063674687	0.0009080479	0.0667822798

P2a Matrix Created (P2 With One Absorbing State) (Columns 14-25)

V14	V15	V16	V17	V18	V19	V20	V21	V22	V23	V24	V25
0.039071354	0.061965637	0.017567251	0.022843437	0.05586194	0.0131803043	0.0008406411	0.018388617	0.08016981	0.0102998857	0.051395361	0.057262552
0.036241042	0.068671876	0.046052872	0.023286503	0.02653367	0.0594907171	0.0314025615	0.024100432	0.08453957	0.0763154026	0.011397804	0.007431698
0.036343610	0.015634060	0.058254671	0.068940020	0.03502252	0.0530351692	0.0538208120	0.034786464	0.07177067	0.0065111489	0.015593101	0.020163484
0.064903229	0.073992678	0.048578745	0.039079231	0.03387034	0.0230652142	0.0123126826	0.007308715	0.05955226	0.0184620897	0.010051536	0.073863669
0.039437891	0.006641166	0.003169252	0.092887474	0.05937934	0.0001944352	0.0211960882	0.070353779	0.08644135	0.0487282660	0.073305014	0.001027014
0.020263486	0.004424652	0.012272275	0.001202473	0.01797174	0.0443074114	0.0368924500	0.026112631	0.04230348	0.0573432963	0.021465728	0.089624364
0.038830488	0.075241580	0.063517388	0.025993456	0.07980768	0.0679881158	0.0337167307	0.030327906	0.00336682	0.0154511801	0.060463716	0.045597923
0.072349223	0.008570857	0.021845412	0.079078120	0.01644535	0.0586633036	0.0066868954	0.064423497	0.04292341	0.0268640088	0.011930653	0.028626842
0.014598340	0.030623183	0.059030829	0.001586694	0.06238343	0.0103549218	0.0640858798	0.065211921	0.01553597	0.0448349816	0.035463825	0.075579723
0.000000000	0.000000000	0.000000000	0.000000000	0.00000000	0.0000000000	0.0000000000	0.000000000	0.00000000	0.0000000000	0.000000000	0.000000000
0.029595055	0.012002106	0.040809434	0.069327102	0.05229019	0.0196233188	0.0577955370	0.071963952	0.01135631	0.0210999419	0.052796175	0.046667666
0.063037903	0.029224594	0.047903288	0.064325651	0.03818162	0.0402062962	0.0055936045	0.024793344	0.04305982	0.0066082310	0.045060730	0.059480757
0.049069657	0.015221212	0.019039900	0.057582098	0.01979131	0.0579873485	0.0457339681	0.029284249	0.05398049	0.0517585128	0.051466410	0.036799134
0.008719531	0.018483837	0.024942460	0.070449426	0.02839155	0.0246714340	0.0200738033	0.045799920	0.01176957	0.0502605567	0.005551154	0.052530316
0.023506122	0.029071040	0.051810597	0.063656905	0.02856555	0.0213747145	0.0522716312	0.012239827	0.06694585	0.0569015822	0.007537236	0.061691323
0.017863894	0.026841689	0.064164393	0.062484608	0.04038548	0.0512802208	0.0276889313	0.008759161	0.06663501	0.0759642631	0.083259492	0.016316094
0.037237279	0.061907234	0.082779099	0.053322567	0.03718934	0.0382441179	0.0053855382	0.011196424	0.05716297	0.0301907966	0.037032034	0.001959026
0.037444900	0.004888026	0.065567044	0.058508846	0.06800701	0.0694226480	0.0118324701	0.054042435	0.05255097	0.0280339784	0.057733865	0.040970646
0.072510929	0.014370422	0.077046081	0.026323828	0.08076053	0.0519625431	0.0730240961	0.045118646	0.06377891	0.0008786291	0.048365631	0.016413631
0.065464404	0.047608792	0.067851034	0.028336367	0.05845722	0.0746728748	0.0629365575	0.039238627	0.01640061	0.0399394473	0.042101375	0.078020271
0.046461551	0.063601667	0.051102912	0.076618048	0.06161793	0.0274405206	0.0049686379	0.077980290	0.03392165	0.0249267895	0.057927130	0.040945733
0.005062167	0.047079058	0.035574837	0.048269039	0.07670249	0.0388689358	0.0318847061	0.025241010	0.03945621	0.0509735390	0.028692982	0.085199813
0.027740128	0.037276867	0.072635984	0.050047847	0.02256215	0.0352868810	0.0481248120	0.052938686	0.03376354	0.0150607025	0.041276979	0.019951175
0.039284589	0.066428418	0.019854764	0.075223191	0.02375995	0.0460792552	0.0223858828	0.013470713	0.01624876	0.0265632962	0.080186735	0.040116599
0.075402066	0.007319377	0.035978055	0.071443356	0.02810298	0.0662837391	0.0128719037	0.017004229	0.07033906	0.0523438882	0.035661860	0.034051741

NOTE : Since P3 and P3a matrices are of size 50, their pictures are inefficient to be shown here. They can be found in the following drive link:

https://drive.google.com/drive/folders/1w7MmJ9EMGGSIG1x8AYxDtvtTUk1gfNCl?usp=sharing

Monte Carlo Simulation Method Results

```
> MonteP1
[1] 0.257745 0.206920 0.224170 0.089935 0.221230
> MonteP2
 [1] 0.029130 0.034210 0.040625 0.042720 0.040865 0.037585 0.048755 0.035255 0.039260 0.039620 0.042475 0.034835 0.044150
[14] 0.040975 0.035530 0.044255 0.051795 0.045040 0.042210 0.031465 0.036920 0.044315 0.034450 0.040715 0.042845
> MonteP3
 [1] 0.018610 0.021565 0.021850 0.019450 0.018915 0.018790 0.017445 0.018530 0.019795 0.018200 0.019505 0.019445 0.020740
[14] 0.021405 0.021400 0.019315 0.021485 0.020515 0.018830 0.025765 0.023115 0.017820 0.018640 0.018780 0.019660 0.019915
[27] 0.023545 0.019975 0.022385 0.022575 0.021365 0.020420 0.018145 0.019205 0.023300 0.015250 0.019890 0.018555 0.021285
[40] 0.020540 0.017630 0.019550 0.022285 0.019730 0.018755 0.019585 0.018800 0.017540 0.018390 0.021815
```

Matrix Multiplication Method Results

```
> MultiP1
[1] 0.25852277 0.20748776 0.22290885 0.08925351 0.22182711
> MultiP2
 [1] 0.02880336 0.03466053 0.04024950 0.04289424 0.04023213 0.03815806 0.04884253 0.03524360 0.03918513 0.03901560 0.04360401
[12] 0.03374622 0.04471582 0.04093041 0.03558107 0.04436113 0.05094293 0.04544573 0.04256949 0.03164305 0.03631296 0.04463027
[23] 0.03459631 0.04058166 0.04305425
> MultiP3
 [1] 0.01808232 0.02151265 0.02188198 0.01981494 0.01824810 0.01964560 0.01739568 0.01892259 0.01987958 0.01834013 0.01922392
[12] 0.01958788 0.02064680 0.02152369 0.02105053 0.01927201 0.02151316 0.02054972 0.01912388 0.02600987 0.02303399 0.01785369
[23] 0.01863600 0.01865003 0.01987699 0.01977093 0.02338291 0.01995333 0.02174200 0.02243226 0.02131833 0.02052144 0.01754157
[34] 0.01935532 0.02377449 0.01526805 0.01996691 0.01855038 0.02132626 0.02068447 0.01760774 0.01954579 0.02176782 0.01997692
[45] 0.01892231 0.01952362 0.01864342 0.01768211 0.01824633 0.02221958
```

Monte Carlo Simulation Method Results (With Absorbing States)

```
> MonteP1a
[1] 0.000010 0.000005 0.999975 0.000000 0.000010
> MonteP2a
 [1] 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
> MonteP3a
 [1] 0.000005 0.000000 0.000005 0.000005 0.000005 0.999825 0.000000 0.000005 0.000005 0.000000 0.000005 0.000000 0.000000 0.000005 0.000015
[16] 0.000005 0.000010 0.000005 0.000000 0.000005 0.000005 0.000005 0.000000 0.000005 0.000010 0.000005 0.000005 0.000000 0.000005 0.000000
[31] 0.000005 0.000005 0.000010 0.000000 0.000005 0.000000 0.000005 0.000000 0.000000 0.000005 0.000010 0.000000 0.000005 0.000000 0.000005
[46] 0.000000 0.000000 0.000000 0.000000 0.000000
```

Matrix Multiplication Method Results (With Absorbing States)

```
> MultiP1a
[1] 4.393939e-05 3.864998e-05 9.998606e-01 1.964617e-05 3.716915e-05
> MultiP2a
 [1] 0.0001647276 0.0001996826 0.0002222807 0.0002324169 0.0002234389 0.0002121022 0.0002744200 0.0002022681 0.0002154800 0.9946426170
[11] 0.0002435217 0.0001794267 0.0002577699 0.0002202915 0.0001920317 0.0002562393 0.0002882804 0.0002477478 0.0002341644 0.0001727979
[21] 0.0001995986 0.0002594478 0.0001962316 0.0002288108 0.0002342057
> MultiP3a
 [1] 1.146905e-04 1.326505e-04 1.373044e-04 1.217971e-04 1.154753e-04 9.938789e-01 1.082825e-04 1.163767e-04 1.221684e-04 1.134642e-04
[11] 1.214844e-04 1.205592e-04 1.302311e-04 1.330924e-04 1.299526e-04 1.197066e-04 1.328667e-04 1.286372e-04 1.191623e-04 1.635897e-04
[21] 1.425240e-04 1.118964e-04 1.163476e-04 1.181510e-04 1.259459e-04 1.241407e-04 1.478205e-04 1.260014e-04 1.353620e-04 1.404053e-04
[31] 1.347772e-04 1.275076e-04 1.099424e-04 1.211932e-04 1.471511e-04 9.622441e-05 1.248183e-04 1.158097e-04 1.356152e-04 1.282410e-04
[41] 1.120782e-04 1.238196e-04 1.345756e-04 1.229207e-04 1.186657e-04 1.208162e-04 1.142646e-04 1.125073e-04 1.123886e-04 1.376543e-04
```