Stochastic Processes: Final Exam

Martín Prado

June 4, 2024

Universidad de los Andes − Bogotá Colombia

Exercise 1

Consider a compound Poisson process with rate λ and increment distribution μ.

Calculate the characteristic function, the exponential moments (if finite), the mean, the variance and the density (in the simplest form) for

(a) $\lambda = 1,\ \mu = \text{EXP}(1)$.

(b) $\lambda > 0,\ \mu = \text{EXP}(\kappa), \kappa > 0$.

(c) $\lambda = 1,\ \mu = N(0,1)$.

(d) $\lambda > 0,\ \mu = N(m, \sigma^2), m \in \mathbb{R}, \sigma > 0$.

(e) $\lambda = 1,\ \mu = \text{Geo}(\frac{1}{2})$.

(f) $\lambda > 0,\ \mu = \text{Geo}(p)$.

Solution Characteristic Function

Let $Z \sim \mu$ and $N(t) \sim \text{Pois}(\lambda t)$. According to Lemma 4.4 from the notes, if $C \sim \text{CPP}(\lambda t, \mu)$, then its characteristic function is

$$\phi_{C_t}(z) = \mathbf{E}\left[e^{izC_t}\right] = e^{\lambda t(\phi_Z(z)-1)}.$$

Thus, in every case, this function should be

(a) The characteristic function for the exponential distribution $Z \sim \text{EXP}(\kappa)$ is

$$\phi_Z(z) = (1 - iz\kappa^{-1})^{-1}.$$

Thus, for $\kappa = 1$ and $\lambda = 1$,

$$\phi_{C_t}(z) = \exp\left[t \cdot \left(\frac{1}{1-iz} - 1\right)\right] = \exp\left(t \cdot \frac{iz}{1-iz}\right).$$

(b) Now, in general, for $\kappa > 0$ and $\lambda > 0$,

$$\phi_{C_t}(t) = \exp\left[\lambda t \left(\frac{1}{1-iz\kappa^{-1}} - 1\right)\right] = \exp\left(\lambda t \cdot \frac{iz\kappa^{-1}}{1-iz\kappa^{-1}}\right).$$

(c) The characteristic function for the Normal distribution $Z \sim N(m, \sigma^2)$ is

$$\phi_Z(z) = \exp\left(izm - \frac{1}{2}\sigma^2 z^2\right).$$

Thus, for $\lambda = 1$, $m = 0$ and $\sigma^2 = 1$,

$$\phi_{C_t}(t) = \exp\left[t(e^{-\frac{1}{2}z^2} - 1)\right].$$

(d) Now, in general, for $m \in \mathbb{R}$, $\sigma > 0$ and $\lambda > 0$,

$$\phi_{C_t}(t) = \exp\left[\lambda t \cdot \exp\left(izm - \frac{1}{2}\sigma^2 z^2\right) - \lambda t\right].$$

(e) The characteristic function for the geometric distribution $Z \sim \text{Geo}(p)$ is

$$\phi_Z(z) = \frac{p}{e^{-iz} - (1-p)}$$

Thus, for $p = \frac{1}{2}$ and $\lambda = 1$,

$$\phi_{C_t}(t) = \exp\left(t \cdot \frac{1/2}{e^{-iz} - 1/2} - t\right) = \exp\left(t \cdot \frac{1 - e^{-iz}}{e^{-iz} - 1/2}\right).$$

(f) Now, in general, for $0 < p < 1$ and $\lambda > 0$,

$$\phi_{C_t}(t) = \exp\left[\lambda t \cdot \left(\frac{p}{e^{-iz} - (1-p)} - 1\right)\right].$$

Solution Exponential Moment

Note that in the proof of $\phi_{C_t}(t) = e^{\lambda t(\phi_Z(z)-1)}$, the fact that e^{iz} was a complex number. Thus, for the moment generating function

$$M_{C_t}(t) = e^{\lambda t(M_Z(z)-1)}$$

The same principle applies, but replacing iz by just z.

(a) $\exp\left(t \cdot \frac{z}{1-z}\right)$

(b) $\exp\left(\lambda t \cdot \frac{z\kappa^{-1}}{1-z\kappa^{-1}}\right)$

(c) $\exp\left[t(e^{\frac{1}{2}z^2} - 1)\right]$ (The negative sign becomes positive)

(d) $\exp\left[\lambda t \cdot \exp\left(zm + \frac{1}{2}\sigma^2 z^2\right) - \lambda t\right]$

(e) $\exp\left(t \cdot \frac{1-e^{-z}}{e^{-z}-1/2}\right)$

(f) $\exp\left[\lambda t \cdot \left(\frac{p}{e^{-z}-(1-p)} - 1\right)\right]$

Solution Expected Value

We know that if $C_t \sim \text{CPP}(\lambda t, \mu)$, then there is a random variable $N(t) \sim \text{Pois}(\lambda t)$ and a sequence of i.i.d. random variables (independent from N) $Z_k \sim \mu$ such that

$$C_t = \sum_{k=1}^{N(t)} Z_k.$$

Finally, the law of total expectation implies

$$\mathbf{E}\left[C_t\right] = \mathbf{E}\left[\mathbf{E}\left[C_t \mid N(t)\right]\right] = \mathbf{E}\left[N(t) \cdot \mathbf{E}\left[Z\right]\right] = \mathbf{E}\left[N(t)\right] \cdot \mathbf{E}\left[Z\right] = \lambda t \cdot \mathbf{E}\left[Z\right].$$

For every item, this expected value is:

(a) The expected value $\mathbf{E}\left[Z\right]$ for $Z \sim \text{EXP}(\kappa)$ is κ^{-1}. Thus, for $\kappa = 1$ and $\lambda = 1$,

$$\mathbf{E}\left[C_t\right] = \lambda t \cdot \mathbf{E}\left[Z\right] = t \cdot 1^{-1} = t.$$

(b) For the general case,

$$\mathbf{E}\left[C_t\right] = \lambda t \cdot \mathbf{E}\left[Z\right] = \lambda t \cdot \kappa^{-1}.$$

(c) The expected value $\mathbf{E}\left[Z\right]$ for $Z \sim N(m, \sigma^2)$ is m. Thus, for $m = 0$, $\sigma^2 = 1$ and $\lambda = 1$,

$$\mathbf{E}\left[C_t\right] = \lambda t \cdot \mathbf{E}\left[Z\right] = \lambda t \cdot m = 0.$$

(d) For the general case,

$$\mathbf{E}\left[C_t\right] = \lambda t \cdot \mathbf{E}\left[Z\right] = \lambda t \cdot m.$$

(e) The expected value $\mathbf{E}\left[Z\right]$ for $Z \sim \text{Geo}(p)$ is p^{-1}. Thus, for $p = 1/2$ and $\lambda = 1$,

$$\mathbf{E}\left[C_t\right] = \lambda t \cdot \mathbf{E}\left[Z\right] = t \cdot (1/2)^{-1} = 2t.$$

(f) For the general case,

$$\mathbf{E}\left[C_t\right] = \lambda t \cdot \mathbf{E}\left[Z\right] = \lambda t \cdot p^{-1}.$$

Solution Variance

Remember that for a constant $c \in \mathbb{R}$,

$$\mathbf{Var}\left[cN\right] = c^2 \mathbf{Var}\left(N\right),$$

thus,

$$\mathbf{Var}\left[N \cdot \mathbf{E}\left[Z\right]\right] = \mathbf{E}\left[Z\right]^2 \cdot \mathbf{Var}\left[N\right]$$

The law of total variance states that

$$\begin{aligned}
\mathbf{Var}\left[C_t\right] &= \mathbf{E}\left[\mathbf{Var}\left[C_t \mid N(t)\right]\right] + \mathbf{Var}\left[\mathbf{E}\left[C_t \mid N(t)\right]\right]. \\
&= \mathbf{E}\left[N(t) \cdot \mathbf{Var}\left[Z\right]\right] + \mathbf{Var}\left[N(t) \cdot \mathbf{E}\left[Z\right]\right] \\
&= \mathbf{E}\left[N(t)\right] \cdot \mathbf{Var}\left[Z\right] + \mathbf{E}\left[Z\right]^2 \cdot \mathbf{Var}\left[N(t)\right] \\
&= \lambda t \cdot \mathbf{Var}\left[Z\right] + \lambda t \cdot \mathbf{E}\left[Z\right]^2.
\end{aligned}$$

Now, for every item, the variance is:

(a) The variance $\mathbf{Var}\left[Z\right]$ for $Z \sim \mathrm{EXP}(\kappa)$ is κ^{-2}. Thus, for $\kappa = 1$ and $\lambda = 1$,

$$\mathbf{Var}\left[C_t\right] = \lambda t \cdot \mathbf{Var}\left[Z\right] + \lambda t \cdot \mathbf{E}\left[Z\right]^2 = t \cdot 1^{-2} + t \cdot (1^{-1})^2 = 2t.$$

(b) For the general case,

$$\mathbf{Var}\left[C_t\right] = \lambda t \cdot \mathbf{Var}\left[Z\right] + \lambda t \cdot \mathbf{E}\left[Z\right]^2 = \lambda t \cdot \kappa^{-2} + \lambda t(\kappa^{-1})^2 = 2\lambda t \cdot \kappa^{-2}.$$

(c) The variance $\mathbf{Var}\left[Z\right]$ for $Z \sim N(m, \sigma^2)$ is σ^2. Thus, for $m = 0$, $\sigma^2 = 1$ and $\lambda = 1$,

$$\mathbf{Var}\left[C_t\right] = \lambda t \cdot \mathbf{Var}\left[Z\right] + \lambda t \cdot \mathbf{E}\left[Z\right]^2 = t \cdot 1 + 0 = t.$$

(d) For the general case,

$$\mathbf{Var}\left[C_t\right] = \lambda t \cdot \mathbf{Var}\left[Z\right] + \lambda t \cdot \mathbf{E}\left[Z\right]^2 = \lambda t \cdot \sigma^2 + \lambda t \cdot m^2.$$

(e) The variance $\mathbf{Var}\left[Z\right]$ for $Z \sim \mathrm{Geo}(p)$ is $\frac{1-p}{p^2}$. Thus, for $p = 1/2$ and $\lambda = 1$,

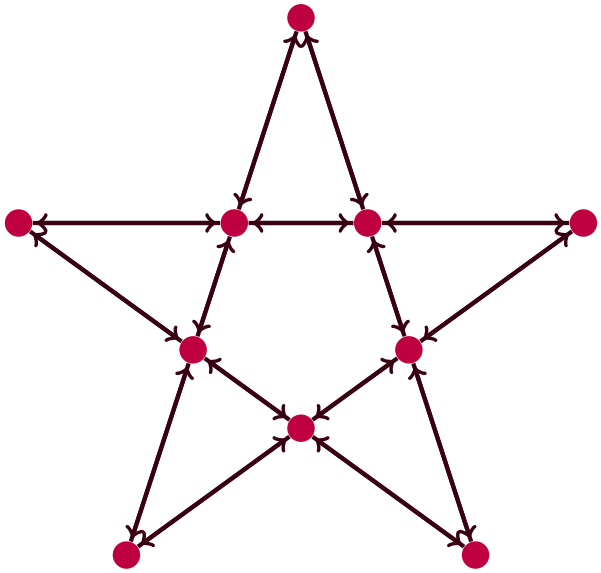
$$\mathbf{Var}\left[C_t\right] = \lambda t \cdot \mathbf{Var}\left[Z\right] + \lambda t \cdot \mathbf{E}\left[Z\right]^2 = t \cdot (1/2)^{-1} + t \cdot (1/2)^{-2} = 6t.$$

(f) For the general case,

$$\mathbf{Var}\left[C_t\right] = \lambda t \cdot \mathbf{Var}\left[Z\right] + \lambda t \cdot \mathbf{E}\left[Z\right]^2 = \lambda t \cdot \frac{1-p}{p^2} + \lambda t \cdot p^{-2} = \lambda t \cdot p^{-2}(2-p).$$

Exercise 2

Consider the following 10 vertex graph and a random walk over these states



(a) Enumerate the states i by $\{0, \ldots, 9\}$ and assign to the time of the state exit, the rate $i+1$. Construct the Q-matrix of the resulting continuous time Markov chain.

(b) Calculate the invariant measure of this continuous time Markov chain.

(c) Build the transition matrix of the resulting discrete Markov chain.

(d) Calculate the invariant measure and compare with the measure of item (b).

(e) For item 2, simulate 100 trajectories of $[0, 100]$ and approximate the invariant measure by the histograms of the visit period of each trajectory. Graph the trajectories, the histogram and the histogram of the visit times of each state.

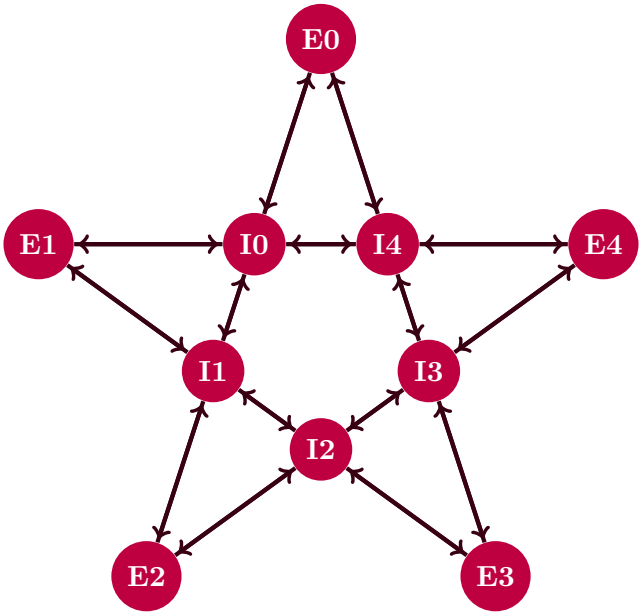
(f) For item 4, simulate 100 trajectories of $[0, 100]$ and approximate the invariant measure by the histograms of the visit period of each trajectory. Graph the trajectories, the histogram and the histogram of the visit times of each state.

Solution Part (a)

Note: For this solution, we are going to start counting from $i = 0$, and thus, the rate of the state i is going to be $i + 1$.

Beforehand, for the simulations and calculations, we are going to use Python with the following packages:

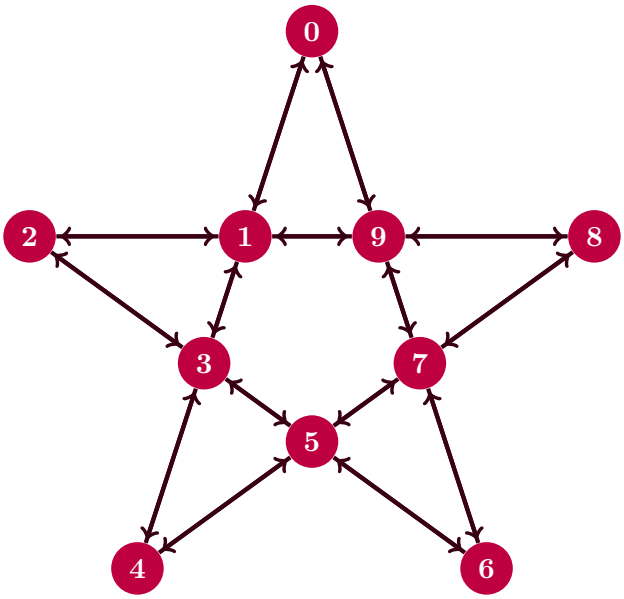
```
import sympy as sp
import numpy as np
import scipy as sci
import matplotlib.pyplot as plt
```



The way we assign these numbers is arbitrary. The only important rules for the adjacency of this graph are, for $i \in \mathbb{Z}_5$:

$$E_i \longleftrightarrow I_i, \quad E_i \longleftrightarrow I_{i-1}, \quad I_i \longleftrightarrow I_{i-1}.$$

However, for this exercise, we are going to enumerate the graph as follows:




```python
exterior = [0,2,4,6,8]
interior = [1,3,5,7,9]
states = np.arange(10)
adjacency = np.zeros((10,10))
for i in range(5):
    adjacency[interior[i],exterior[i]] += 1
    adjacency[interior[i],exterior[(i-1)%5]] += 1
    adjacency[interior[i],interior[(i-1)%5]] += 1
adjacency += adjacency.T
```

$$\text{Adj} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Since there is no loop in the graph, the jump matrix (Denoted by Π) of this process can be is obtained from the adjacency matrix with each row being divided by the row sum:

```python
jump_matrix = adjacency / adjacency.sum(axis=1)[:,None]
```

$$\Pi = \begin{bmatrix} 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\ \frac{1}{4} & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{4} & \frac{1}{4} & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & \frac{1}{4} & \frac{1}{4} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 \end{bmatrix}$$

For the generator matrix of this process (denoted by Q), note that the holding rates, according to the exercise, are $q_i = i + 1$ for the state i. Thus, the diagonal of this matrix should be $-q_i = -(i+1)$. The other entries in each row have equal weight, so $q_{i,j} = \pi_{i,j}/q_i$:

```
holding_rates = np.array([i+1 for i in range(10)])
Q_matrix = -np.diag(holding_rates).astype(float)
Q_matrix += adjacency * (holding_rates / adjacency.sum(axis=1))[:,None]
```

$$Q = \begin{bmatrix} -1 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\ \frac{1}{2} & -2 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\ 0 & \frac{3}{2} & -3 & \frac{3}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & -4 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{5}{2} & -5 & \frac{5}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{3}{2} & \frac{3}{2} & -6 & \frac{3}{2} & \frac{3}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{7}{2} & -7 & \frac{7}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 2 & -8 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{9}{2} & -9 & \frac{9}{2} \\ \frac{5}{2} & \frac{5}{2} & 0 & 0 & 0 & 0 & 0 & \frac{5}{2} & \frac{5}{2} & -10 \end{bmatrix}$$

Solution Part (b)

In order to find the left eigenvector associated with an specific eigenvalue, we designed the following function:

```
def get_left_eigenvector(matrix, desired_eigenvalue):
    eigenvalues, multiplicity, eigenvectors = list(zip(*sp.Matrix(matrix.T).eigenvects()))

    solution_index = np.argmin(np.abs(desired_eigenvalue - np.array(eigenvalues)))
    solution = eigenvectors[solution_index][0]
    solution = solution / sum(solution)
    return sp.nsimplify(solution.T.simplify(), tolerance=0.0001, rational= True)
```

The invariant distribution λ associated to this continuous process can be obtained using

many methods. In fact, it's the solution to the following equations:

$$(1): \quad \mu\Pi = \mu, \quad \lambda_i = \frac{\mu_i/q_i}{\sum_{i=1}^{|I|} \mu_i/q_i}$$

$$(2): \quad \lambda Q = 0$$

$$(3): \quad \lambda P(t) = \lambda, \quad P(t) = e^{tQ}, \ \forall t \geq 0$$

Method (1)

For method (1), we obtain

```
mu_invariant = get_left_eigenvector(jump_matrix, 1)
```

$$\mu = \begin{bmatrix} \frac{1}{15} & \frac{2}{15} & \frac{1}{15} & \frac{2}{15} & \frac{1}{15} & \frac{2}{15} & \frac{1}{15} & \frac{2}{15} & \frac{1}{15} & \frac{2}{15} \end{bmatrix}$$

```
lambda_invariant_1 = mu_invariant/holding_rates
lambda_invariant_1 = lambda_invariant_1/lambda_invariant_1.sum()
```

$$\lambda_1 = \begin{bmatrix} \frac{1260}{5129} & \frac{1260}{5129} & \frac{420}{5129} & \frac{630}{5129} & \frac{252}{5129} & \frac{420}{5129} & \frac{180}{5129} & \frac{315}{5129} & \frac{140}{5129} & \frac{252}{5129} \end{bmatrix}.$$

Method (2)

Then, for the next method, we have

```
lambda_invariant_2 = get_left_eigenvector(Q_matrix, 0)
```

$$\lambda_2 = \begin{bmatrix} \frac{1260}{5129} & \frac{1260}{5129} & \frac{420}{5129} & \frac{630}{5129} & \frac{252}{5129} & \frac{420}{5129} & \frac{180}{5129} & \frac{315}{5129} & \frac{140}{5129} & \frac{252}{5129} \end{bmatrix}.$$

Method (3)

Finally, for the last method, we set $t = 1$ to calculate $P(1) = e^{Q}$

```
P_matrix = sci.linalg.expm(Q_matrix)
lambda_invariant_3 = get_left_eigenvector(P_matrix, 1)
```

Yet again, we obtain the same results:

$$\lambda_3 = \begin{bmatrix} \frac{1260}{5129} & \frac{1260}{5129} & \frac{420}{5129} & \frac{630}{5129} & \frac{252}{5129} & \frac{420}{5129} & \frac{180}{5129} & \frac{315}{5129} & \frac{140}{5129} & \frac{252}{5129} \end{bmatrix}.$$

Therefore, we can be sure that the invariant distribution of this process is

$$\begin{aligned} \lambda &= \begin{bmatrix} \frac{1260}{5129} & \frac{1260}{5129} & \frac{420}{5129} & \frac{630}{5129} & \frac{252}{5129} & \frac{420}{5129} & \frac{180}{5129} & \frac{315}{5129} & \frac{140}{5129} & \frac{252}{5129} \end{bmatrix} \\ &\approx [0.246 \;\; 0.246 \;\; 0.082 \;\; 0.123 \;\; 0.049 \;\; 0.082 \;\; 0.035 \;\; 0.061 \;\; 0.027 \;\; 0.049]. \end{aligned}$$

Solution Part (c)

The matrix the exercise is asking for has diagonal $\gamma_{ii} = 1 - \frac{1}{q_i}$. That is because a success, which is to exit state i, has probability $\frac{1}{q_i}$. Therefore, to exit a given state, has distribution Geo($\frac{1}{q_i}$). It follows that the other row entries (excluding the diagonal) should collectively sum $\frac{1}{q_i}$:

$$\sum_{j \in I} \gamma_{ij} = \gamma_{ii} + \sum_{j \neq i} \gamma_{ij} = 1$$

$$\implies \sum_{j \neq i} \gamma_{ij} = \frac{1}{q_i}.$$

For these probabilities to be correctly weighted according to the graph, the closed formula for γ_{ij} is

$$\gamma_{ij} = \pi_{ij} \cdot \frac{1}{q_i}.$$

Therefore,

```
transition_matrix = np.diag(1-1/holding_rates)
transition_matrix += jump_matrix * (1/holding_rates)[:,None]
```

$$\Gamma = \begin{bmatrix} 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\ \frac{1}{8} & \frac{1}{2} & \frac{1}{8} & \frac{1}{8} & 0 & 0 & 0 & 0 & 0 & \frac{1}{8} \\ 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{16} & \frac{1}{16} & \frac{3}{4} & \frac{1}{16} & \frac{1}{16} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{10} & \frac{4}{5} & \frac{1}{10} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{24} & \frac{1}{24} & \frac{5}{6} & \frac{1}{24} & \frac{1}{24} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{14} & \frac{6}{7} & \frac{1}{14} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{32} & \frac{1}{32} & \frac{7}{8} & \frac{1}{32} & \frac{1}{32} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{18} & \frac{8}{9} & \frac{1}{18} \\ \frac{1}{40} & \frac{1}{40} & 0 & 0 & 0 & 0 & 0 & \frac{1}{40} & \frac{1}{40} & \frac{9}{10} \end{bmatrix}$$

Solution Part (d)

The invariant distribution ν for this process is the solution of the following system

$$\nu\Gamma = \nu.$$

```
transition_invariant = get_left_eigenvector(transition_matrix, 1)
```

$$\begin{aligned}\nu &= \left[\tfrac{1}{85} \quad \tfrac{4}{85} \quad \tfrac{3}{85} \quad \tfrac{8}{85} \quad \tfrac{1}{17} \quad \tfrac{12}{85} \quad \tfrac{7}{85} \quad \tfrac{16}{85} \quad \tfrac{9}{85} \quad \tfrac{4}{17}\right]. \\ &\approx [0.012\ 0.047\ 0.035\ 0.094\ 0.059\ 0.141\ 0.082\ 0.188\ 0.106\ 0.235]\end{aligned}$$

This measure seems to have no relation with the other process

Solution Part (e)

We have a sequence $(X^j, T^j)_{j\in\mathbb{N}}$ of steps in one trajectory. Assume that m is the total number of jumps that the process makes before stopping at the time T. Then, for such trajectory we can define X_t as follows

$$0 = T^0 < \cdots < T^m \leq T,$$

$$X_t = X^j, \quad \text{for } t \in [T^j, T^{j+1})$$

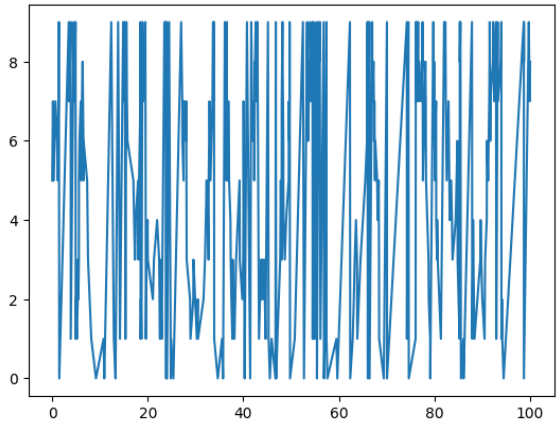
$$\mathbf{P}\{X^{j+1} = l \mid X^j = k, \ldots, X^0 = k_0\} = \mathbf{P}\{X^{j+1} = l \mid X^j = k\} = \pi_{kl}.$$

The last property is the Markov property. Finally, T^k is independent from X^l for every $k, l \in \mathbb{N}$, and $T^j \sim \text{Exp}(q_{X^j})$. We are going to simulate this process N times:

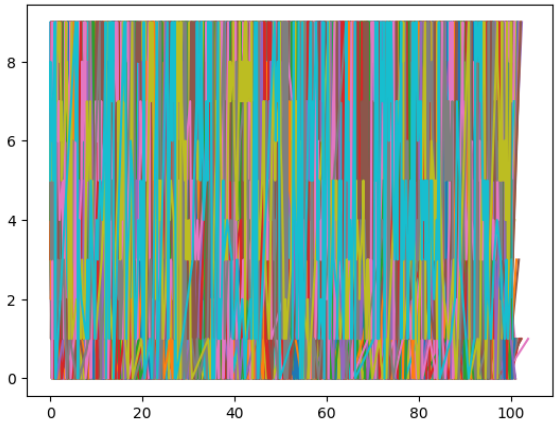
```
T = 100 # max time
N = 100 # number of trajectories
trajectories = []

for trajectory_index in range(N): # simulating N trajectories
    current_time = 0
    time_list = [current_time]
    state_list = [np.random.choice(states, p=mu_invariant)]

    while current_time < T:
        current_state = state_list[-1]
        current_rate = holding_rates[current_state]
        current_time += np.random.exponential(scale=1/holding_rates[current_state])
        time_list.append(current_time)
        next_state = np.random.choice(states, p=jump_matrix[current_state])
        state_list.append(next_state)
```

```
        state_list = np.array(state_list)
        time_list = np.array(time_list)
        trajectories.append((time_list, state_list))
```

Note: We started the simulation with $X_0 \sim \mu$. The results are the following,



1 trajectory graph



The graph mess for all the trajectories

Estimating μ with the frequencies

Our estimator for μ is going to be

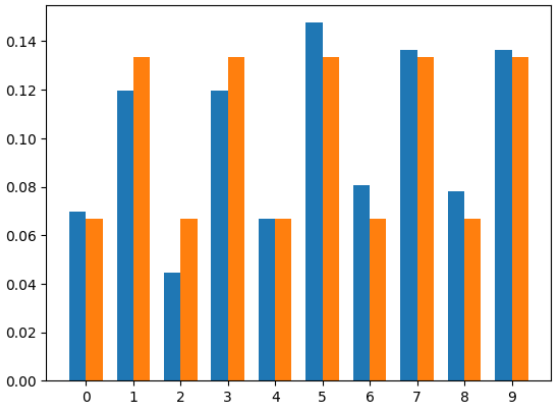
$$\widehat{\mu}_i = \frac{1}{m} \sum_{j=0}^{m} \mathbb{1}(X^j = i).$$

This is the average number of times that our process jumps to the state i, and since the Markov process X^j is independent from T^j, it follows from the ergodic theorem that $\widehat{\mu}_i \to \mu$ as m goes to infinity.

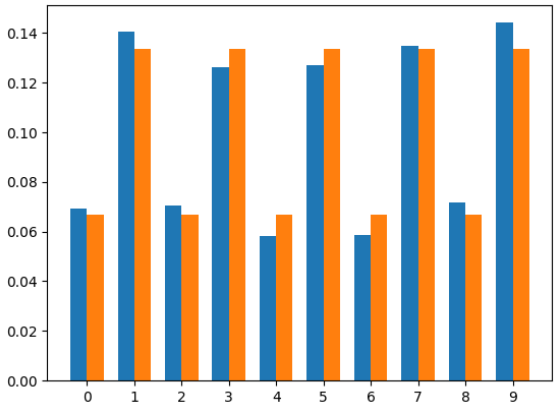
```
    def get_visit_frequency(trajectory):
        trajectory = trajectories[0]
        time_list, state_list = trajectory
        state_list = state_list[:-1]
        values, indices, counts = np.unique(state_list[state_list.argsort()],
            return_counts=True, return_index=True)

        visit_frequency = counts/counts.sum()
        return visit_frequency
```

To produce a comparison, we make an average of this visit frequency from the list of all the trajectories,

```
visit_frequencies = np.array([get_visit_frequency(trajectory)
    for trajectory in trajectories])
average_visit_frequency = visit_frequencies.mean(axis = 0)

bar_width = 0.35
x = range(len(average_visit_frequency))

fig, ax = plt.subplots()
bar1 = ax.bar(x, average_visit_frequency, bar_width,
    label=r'$\wh{\mu}_i = \frac{1}{m}\sum_{j = 0}^{m} 1(X^j = i)$')
bar2 = ax.bar([i + bar_width for i in x], mu_invariant, bar_width, label=r"$\mu_i$")

ax.set_xticks([i + bar_width / 2 for i in x])
ax.set_xticklabels([str(i) for i in range(len(average_visit_frequency))])
```



$N = 100,\ T = 100$



$N = 100,\ T = 1000$

The blue bar (left) is the result of the estimator $\widehat{\boldsymbol{\mu}}_{\boldsymbol{i}}$ simulated $N = 100$ times.
The orange bar (right) is real value of the invariant measure $\boldsymbol{\mu_i}$.

Estimating q_i with the visit lengths

Let T_i^n be the time of the n-th return to the state i. Also, let M_i^n be the length of the n-th visit to i. In other words,

$$M_i^n = \inf\{t > T_i^n : X_t \neq i\} - T_i^n.$$

According to the ergodic theorem,

$$\frac{M_i^1 + \cdots + M_i^n}{n} \to \frac{1}{q_i}, \text{ as } n \to \infty.$$

Thus, we can propose an estimator

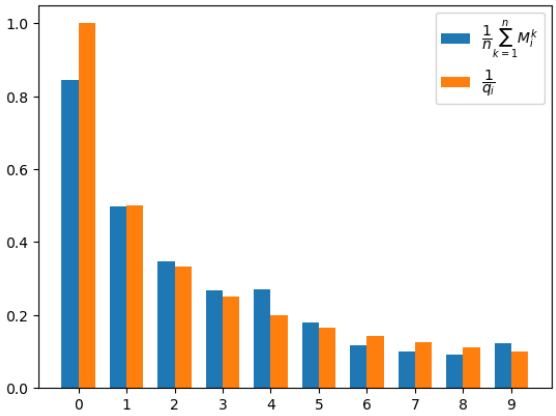
$$\widehat{q}_i = \left(\frac{M_i^1 + \cdots + M_i^{n_i}}{n_i}\right)^{-1},$$

where n_i is the last time that we visit i before T.

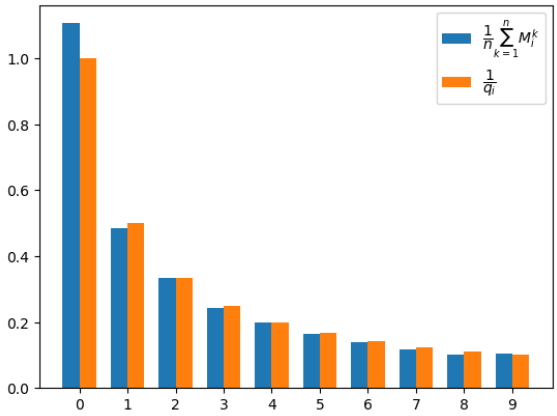
```
def get_average_visit_length(trajectory):
    trajectory = trajectories[0]
    time_list, state_list = trajectory
    state_list = state_list[:-1]
    visit_lengths = time_list[1:] - time_list[:-1]
    values, indices, counts = np.unique(state_list[state_list.argsort()],
        return_counts=True, return_index=True)
    visit_times_grouped = np.split(time_list[:-1][state_list.argsort()], indices)[1:]
    visit_times_grouped = [np.sort(visit_times_grouped[k]) for k in range(10)]
    visit_lengths_grouped = np.split(visit_lengths[state_list.argsort()], indices)[1:]

    average_visit_length = np.array([visit_lengths_grouped[k].mean() for k in range(10)])
    return average_visit_length
```

After making an average of the $N = 100$ simulations of this estimator, we obtain

```
average_visit_length = np.array([get_average_visit_length(trajectory)
    for trajectory in trajectories]).mean(axis = 0)
expected_visit_lengths = [1/holding_rates[k] for k in range(10)]
bar_width = 0.35
x = range(len(average_visit_length))

fig, ax = plt.subplots()
bar1 = ax.bar(x, average_visit_length, bar_width,
    label=r'$\dfrac{1}{n}\sum_{k = 1}^n M_i^k$')
bar2 = ax.bar([i + bar_width for i in x], expected_visit_lengths, bar_width,
    label=r"$\dfrac{1}{q_i}$")

ax.set_xticks([i + bar_width / 2 for i in x])
ax.set_xticklabels([str(i) for i in range(len(average_visit_length))])
ax.legend()
```



$N = 100,\ T = 100$



$N = 100,\ T = 1000$

The blue bar (left) is the result of the estimator $\widehat{q}_i$ simulated $N = 100$ times.
The orange bar (right) is real value of the holding rate q_i.

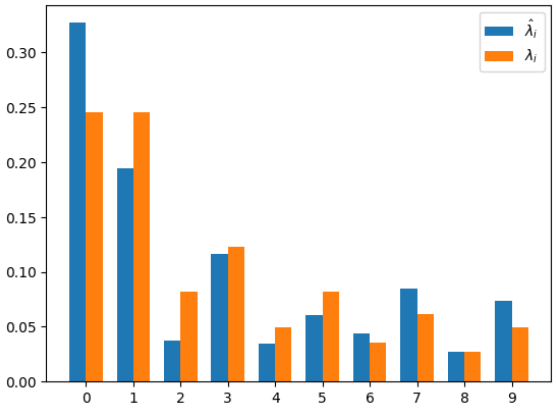
Estimating λ_i with $\widehat{\mu}_i$ and $\widehat{q}_i$

$$\lambda_i = \frac{\mu_i / q_i}{\sum_{i=1}^{|I|} \mu_i / q_i}.$$

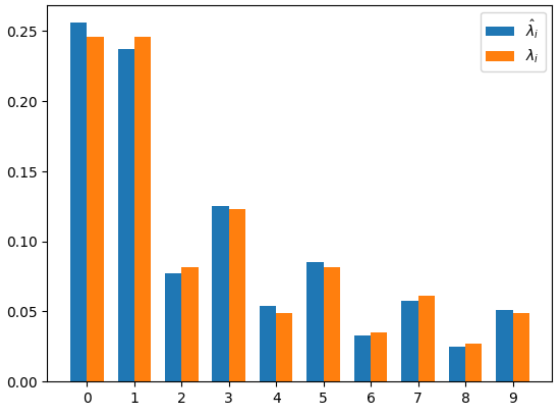
Thus, with the previous results, we can make an estimator of λ with

$$\widehat{\lambda}_i = \frac{\widehat{\mu}_i / \widehat{q}_i}{\sum_{i=1}^{|I|} \widehat{\mu}_i / \widehat{q}_i}$$

```
lambda_estimated_1 = average_visit_frequency*average_visit_length
lambda_estimated_1 = lambda_estimated_1 / lambda_estimated_1.sum()

bar_width = 0.35
x = range(len(lambda_estimated_1))

fig, ax = plt.subplots()
bar1 = ax.bar(x, lambda_estimated_1, bar_width, label=r'$\hat{\lambda}_i$')
bar2 = ax.bar([i + bar_width for i in x], lambda_invariant, bar_width, label=r"$\lambda_i$")

ax.set_xticks([i + bar_width / 2 for i in x])
ax.set_xticklabels([str(i) for i in range(len(lambda_estimated_1))])
ax.legend()
```



$N = 100,\ T = 100$



$N = 100,\ T = 1000$

The blue bar (left) is the result of the estimator $\widehat{\boldsymbol{\lambda}}_{\boldsymbol{i}}$ simulated $N = 100$ times.
The orange bar (right) is real value of the invariant measure $\boldsymbol{\lambda}_{\boldsymbol{i}}$.

My favorite way to estimate λ

We define the n-th excursion time L_i^n to the state i as the difference between return times:

$$L_i^n = T_i^{n+1} - T_i^n$$

According to the ergodic theorem,

$$\widehat{m}_i = \frac{L_i^1 + \cdots + L_i^n}{n} \to m_i = \mathbf{E}\left[T_i\right], \text{ as } n \to \infty.$$

where m_i is the expected return time to the state i. Since $\lambda_i = \frac{1}{m_i q_i}$, we can also produce another estimator

$$\lambda_i^* = \frac{1}{\widehat{m}_i \widehat{q}_i} = \frac{M_i^1 + \cdots + M_i^n}{L_i^1 + \cdots + L_i^n}$$

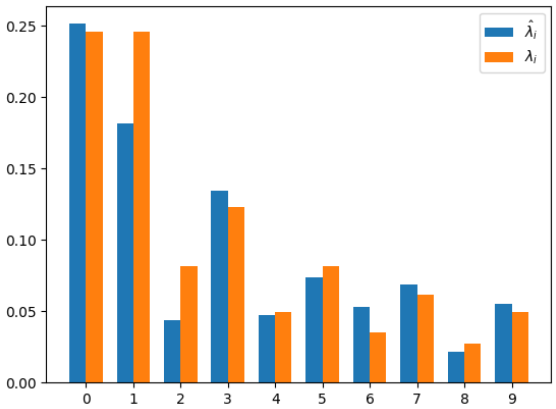
First we define the function for the average excursion times:

```
def get_average_excursion_length(trajectory):
    time_list, state_list = trajectory
    state_list = state_list[:-1]
    visit_lengths = time_list[1:] - time_list[:-1]
    values, indices, counts = np.unique(state_list[state_list.argsort()],
        return_counts=True, return_index=True)

    visit_times_grouped = np.split(time_list[:-1][state_list.argsort()], indices)[1:]
    visit_times_grouped = [np.sort(visit_times_grouped[k])
    for k in range(10)]
```

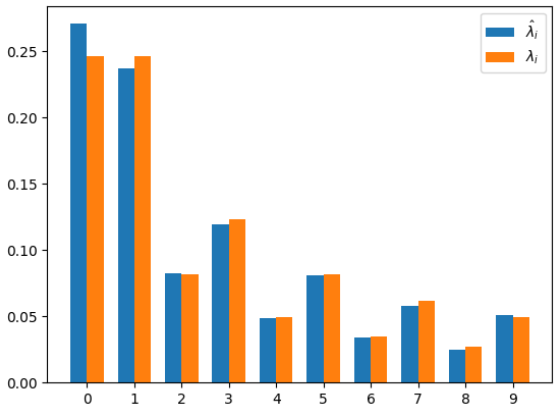
```
        excursion_lengths_grouped = [
            visit_times_grouped[k][1:]- visit_times_grouped[k][:-1] for k in range(10)
            ]

        average_excursion_length = np.array(
            [excursion_lengths_grouped[k].mean() for k in range(10)])
        return average_excursion_length
```

Now we create the estimator and compare it with the real λ

```
    average_excursion_length = np.array(
        [get_average_excursion_length(trajectory) for trajectory in trajectories]
        ).mean(axis = 0)

    lambda_estimated_2 = average_visit_length/average_excursion_length

    bar_width = 0.35
    x = range(len(lambda_estimated_2))

    fig, ax = plt.subplots()
    bar1 = ax.bar(x, lambda_estimated_2, bar_width, label=r'$\hat{\lambda}_i$')
    bar2 = ax.bar([i + bar_width for i in x],
        lambda_invariant, bar_width, label=r"$\lambda_i$")

    ax.set_xticks([i + bar_width / 2 for i in x])
    ax.set_xticklabels([str(i) for i in range(len(lambda_estimated_2))])
    ax.legend()
```



$N = 100,\ T = 100$



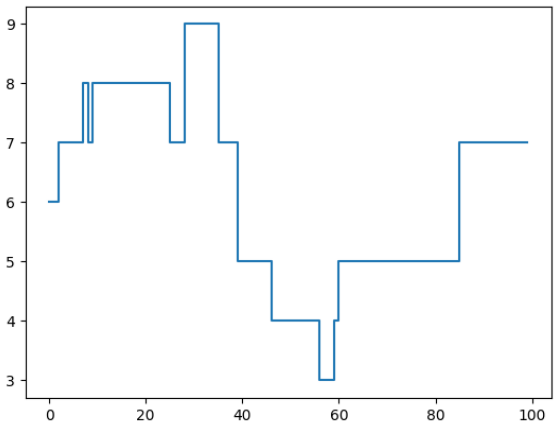
$N = 100,\ T = 1000$

The blue bar (left) is the result of the estimator $\boldsymbol{\lambda}_i^*$ simulated $N = 100$ times.
The orange bar (right) is real value of the invariant measure $\boldsymbol{\lambda_i}$.

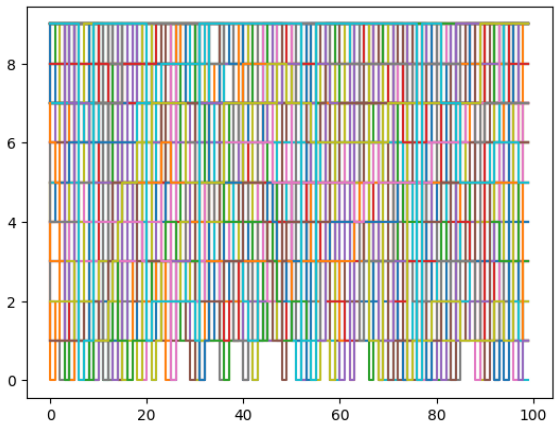
Solution Part (f)

Now, for the discrete process related to the matrix Γ, we simulate $N = 100$ trajectories of $T = 100$ steps

```
T = 100 # max time
N = 100 # number of trajectories
trajectories = []

for trajectory_index in range(N): # simulating N trajectories
    state_list = [np.random.choice(states, p=transition_invariant)]
    for t in range(T):
        current_state = state_list[-1]
        next_state = np.random.choice(states, p=transition_matrix[current_state])
        state_list.append(next_state)
    state_list = np.array(state_list)
    trajectories.append(state_list)
```



1 trajectory graph



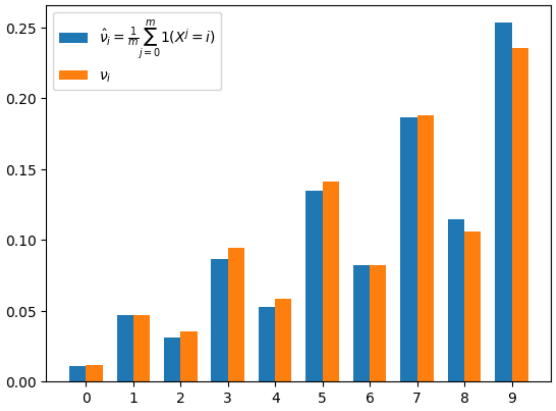
The graph mess for all the trajectories

Finally, we estimate ν the same way we estimated μ previously by taking the frequencies of each state in all the trajectories

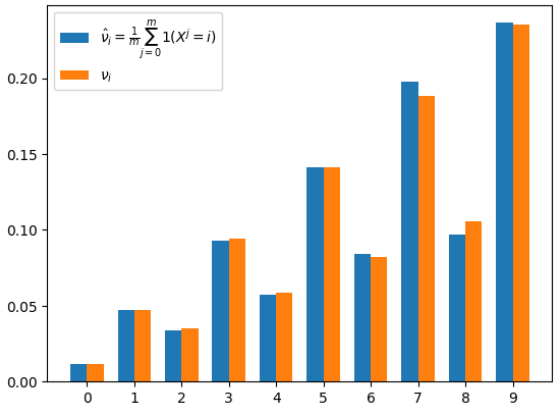
$$\hat{\nu}_i = \frac{1}{m} \sum_{j=0}^{m} \mathbb{1}(X^j = i)$$

```
visit_frequencies = []
for k in range(N):
    state_list = trajectories[k][:-1]
    values, indices, counts = np.unique(state_list[state_list.argsort()], return_counts=True, return_index
    visit_frequency = np.zeros((10,))
```

```
        visit_frequency[values] = counts
        visit_frequencies.append(visit_frequency)
    average_visit_frequency = np.array(visit_frequencies).mean(axis = 0)
    average_visit_frequency = average_visit_frequency/average_visit_frequency.sum()

    bar_width = 0.35
    x = range(len(average_visit_frequency))

    fig, ax = plt.subplots()
    bar1 = ax.bar(x, average_visit_frequency, bar_width, label=r'$\hat{\nu}_i = \frac{1}{m}\sum_{j = 0}^{m} 1(
    bar2 = ax.bar([i + bar_width for i in x], transition_invariant, bar_width, label=r"$\nu_i$")

    ax.set_xticks([i + bar_width / 2 for i in x])
    ax.set_xticklabels([str(i) for i in range(len(average_visit_frequency))])
    ax.legend()
```



$N = 100,\ T = 100$



$N = 100,\ T = 1000$

The blue bar (left) is the result of the estimator $\widehat{\boldsymbol{\nu}}_{\boldsymbol{i}}$ simulated $N = 100$ times.
The orange bar (right) is real value of the invariant measure $\boldsymbol{\nu}_{\boldsymbol{i}}$.