Introduction to Cryptocurrencies (67513)
Exercise 3

Alon Netser
31160253-6

May 17, 2020

1 Mining

Question . Assume two miners are creating blocks, and that the time between blocks is distributed exponentially.
Assume the two miners are creating blocks at rates λ_1, λ_2, respectively. Show, directly from the definition of the exponential distribution, that the probability that miner 1 is the first to create the next block is $\frac{\lambda_1}{\lambda_1+\lambda_2}$.

Solution. Fix a time t (which is continuous, in contrast to the analysis we saw in class). We will look at the probability that miner 1 mined a block (in any time up to time t), but miner 2 didn't. Denote the random-variable which is the time it took miner 1 to mine a block by $X_1 \sim Exp(\lambda_1)$, and the same for miner 2 $X_2 \sim Exp(\lambda_2)$.

Remember that the probability-density-function of an exponential random variable $X \sim Exp(\lambda)$ is $f_X(t) = \lambda e^{-\lambda t}$. So the density functions of X_1 and X_2 are $f_{X_1}(t) = \lambda_1 e^{-\lambda_1 t}$, $f_{X_2}(t) = \lambda_2 e^{-\lambda_2 t}$.

Let's denote the joint probability-density-function of X_1 and X_2 by $f_{X_1,X_2} : A \to \mathbb{R}$, where $A = \{(x_1, x_2) \in \mathbb{R}^2 \mid x_1, x_2 \geq 0\}$. We assume the two random variables are independent, which means that the joint density factorizes to the product of the two (single-variable) density functions, which means that for every $(x_1, x_2) \in \mathbb{R}^2$ it holds that $f_{X_1,X_2}(x_1, x_2) = f_{X_1}(x_1) f_{X_2}(x_2)$. Overall we get that the joint density is $f_{X_1,X_2}(x_1, x_2) = \lambda_1 e^{-\lambda_1 x_1} \lambda_2 e^{-\lambda_2 x_2}$

We are interested in the probability that miner 1 is the first to mine the block, which means that $X_2 \geq X_1$ (note that we can ignore the case where $X_2 = X_1$ because it's a line in $\mathbb{R}^2$ and it has zero measure - it does not affect the probability-mass). In order to calculate this probability-mass, we need to take an integral of f_{X_1,X_2} along the set $B \subseteq A$ which is $B = \{(x_1, x_2) \in \mathbb{R}^2 \mid x_2 > x_1 \geq 0\}$. Note that if we take a look at the first quarter in $\mathbb{R}^2$ and the line $x_1 = x_2$ which divides it into half, the set B is the upper-half ("infinite triangle"). By Fubini's theorem, in order to calculate the integral of f_{X_1,X_2} along the set B, we can

calculate the integrals iteratively. Overall we get that:

$$\begin{aligned}
\Pr[X_2 > X_1] &= \int_B f_{X_1,X_2}(x_1, x_2) dx_1 dx_2 \\
&= \int_0^\infty \int_{x_1}^\infty f_{X_1,X_2}(x_1, x_2) dx_1 dx_2 \\
&= \int_0^\infty \int_{x_1}^\infty \lambda_1 e^{-\lambda_1 x_1} \lambda_2 e^{-\lambda_2 x_2} dx_1 dx_2 \\
&= \int_0^\infty \lambda_1 e^{-\lambda_1 x_1} \left(\int_{x_1}^\infty \lambda_2 e^{-\lambda_2 x_2} dx_2 \right) dx_1 \\
&= \int_0^\infty \lambda_1 e^{-\lambda_1 x_1} \Pr[X_2 > x_1] dx_1 \\
&= \int_0^\infty \lambda_1 e^{-\lambda_1 x_1} (1 - (1 - e^{-\lambda_2 x_1})) dx_1 \\
&= \int_0^\infty \lambda_1 e^{-t(\lambda_1 + \lambda_2)} dt \\
&= \frac{\lambda_1}{\lambda_1 + \lambda_2} \int_0^\infty (\lambda_1 + \lambda_2) e^{-t(\lambda_1 + \lambda_2)} dt \\
&= \frac{\lambda_1}{\lambda_1 + \lambda_2}
\end{aligned}$$

where the last equality is because we are integrating a random exponential variable with rate $(\lambda_1 + \lambda_2)$ from 0 to ∞, which equals 1. Overall, we got that

$$\Pr[\text{miner 1 will be the first to mine a block}] = \Pr[X_2 > X_1] = \frac{\lambda_1}{\lambda_1 + \lambda_2}$$

as required. $\square$

2 vector76

In this part we introduce a new double spend attack that is especially useful when targeting "light nodes", called the **vector76 attack**. Recall, in Bitcoin a "light node", also called an "SPV wallet", is a node that does not download "full" blocks or propagate them to the rest of the network, instead it only downloads block headers. The main difference with respect to the classic double-spend attack we analyzed in class is that the attacker A tricks the light node L into thinking that the chain L currently holds is the "main" chain (the one blocks will be added to), when in fact other honest nodes are not aware of this particular chain. Here is a general description of the attack against a defender that awaits for k confirmations:

- **Phase 1 - Accumulate Confirmations.** A starts mining in secret on a separate branch. In the first block of the secret chain, A places the transaction tx_1 (sending money to the defender) which will later be double spent. The next phase begins once this transaction has k confirmations (blocks built upon it, including the block in which it is included).

- **Phase 2 - Race.** This phase goes on until the attacker's chain is longer than that of the honest nodes (it is possible that this never happens, or that it is immediately true). Once this occurs, showing this chain to the light node L convinces it that the longest chain it sees has at least k-confirmations on tx_1 and that it has received funds. It gives the attacker some item or service in exchange. L does not transmit the blocks in this chain to other nodes, and they remain secret from the rest of the network.

- **Phase 3 - Double spend.** At this point, A transmits tx_2 (which conflicts with tx_1) to the honest nodes who go on to include it in a block, and add to the chain on top of it. The branch secretly mined by the attacker A is never revealed.

Question 1. Describe the success of a vector76 attack in terms of random variables, and compute the probability of success formally and rigorously (recall the classical double-spend attack you saw in class). You may assume that the attacker starts the attack without any pre-mining advantage.

Solution. We follow a similar (but different) argument as done in class. In class, we asked the question "When the **honest-nodes** have k blocks, how many does the **attacker** have?". Now we need to ask the "opposite" question, which is "When the **attacker** has k blocks, how many do the **honest-nodes** have?". This is because in the vector76 attack, the first phase ends when the attacker has k blocks, in contrast to the regular double-spend attack where the first phase ends when the honest-nodes have k blocks. This difference is due to the fact that tx_1 is in the attacker's chain, not in the honest-nodes' chain, so when the attacker will show this chain to the (light-node) defender, he will accept this transaction because it has k confirmations.

We'll denote ℓ_h the amount of blocks the honest-nodes have, and ℓ_a the amount of blocks the attacker has. We'll calculate the probability that the honest-nodes have m blocks, given that the attacker has k blocks. Note that we have $\binom{m+k-1}{m}$ series of $k+m$ blocks, where that last block is the attacker's one (choosing m places for the hones-nodes blocks, within

the first whole blocks excluding the last one). For each such sequence, the probability to get it is $(1-\alpha)^m\alpha^k$ (i.e. that hones-nodes create m blocks, each with probability $1-\alpha$, and the attacker creates k blocks, each with probability α). Overall we get that

$$\Pr[\ell_h = m \mid \ell_a = k] = \binom{m+k-1}{m}(1-\alpha)^m\alpha^k$$

Also notice that for any positive integer k we have

$$1 = \sum_{m=0}^{\infty} \Pr[\ell_h = m \mid \ell_a = k] = \sum_{m=0}^{\infty} \binom{m+k-1}{m}(1-\alpha)^m\alpha^k$$

a fact we'll use later.

Now we move to analyze the second phase of the attack. Once the attacker has k blocks, we calculated the probability that the honest-nodes have m blocks. Now begins the "race to defeat", where both the attacker and the honest-nodes create blocks, and we are interested in the probability that the attacker will ever has more blocks than the honest-nodes. This can be formulated using a Markov chain - we define the random-variable $X_t = \ell_h^t - \ell_a^t$ which is the difference (at time-step t) between the number of blocks the honest-nodes have, and the number of blocks the attacker has. If we ever get to $X_t = -1$ then the attacker wins (he has more blocks than the honest-nodes, and he can now send his secret chain to the light-node L and he'll accept tx_1 as it has at least k confirmations and this chain is longer than the honest-nodes chain). Note that the transition probabilities are the following:

$$\Pr[X_t = n \mid X_{t-1} = n-1] = 1-\alpha$$
$$\Pr[X_t = n \mid X_{t-1} = n+1] = \alpha$$

Now, if the attacker is n blocks behind, the probability he overtakes can be defined using the following recursive formula:

$$\Pr[\text{overtake} \mid X_t = n] = q_n = \begin{cases} 1 & \text{if } n < 0 \\ \alpha q_{n-1} + (1-\alpha)q_{n+1} & \text{if } n \geq 0 \end{cases}$$

The solution to this recursion formula (for $\alpha < 0.5$), as shown in class, is

$$q_n = \begin{cases} 1 & \text{if } n < 0 \\ \left(\frac{\alpha}{1-\alpha}\right)^{n+1} & \text{if } n \geq 0 \end{cases}$$

Putting it all together we get that

$$
\begin{aligned}
\Pr[\text{overtake} \mid \ell_a = k] &= \sum_{m=0}^{\infty} \Pr[\text{overtake} \mid \ell_h = m] \Pr[\ell_h = m \mid \ell_a = k] \\
&= \sum_{m=0}^{\infty} q_{m-k} \binom{m+k-1}{m} (1-\alpha)^m \alpha^k \\
&= \sum_{m=0}^{k-1} 1 \cdot \binom{m+k-1}{m} (1-\alpha)^m \alpha^k \\
&+ \sum_{m=k}^{\infty} (\frac{\alpha}{1-\alpha})^{(m-k)+1} \binom{m+k-1}{m} (1-\alpha)^m \alpha^k \\
&= \sum_{m=0}^{k-1} \binom{m+k-1}{m} (1-\alpha)^m \alpha^k + \sum_{m=k}^{\infty} \binom{m+k-1}{m} (1-\alpha)^{k-1} \alpha^{m+1} \\
&= \sum_{m=0}^{k-1} \binom{m+k-1}{m} (1-\alpha)^m \alpha^k + \frac{\alpha}{1-\alpha} \sum_{m=k}^{\infty} \binom{m+k-1}{m} (1-\alpha)^k \alpha^m \\
&= \sum_{m=0}^{k-1} \binom{m+k-1}{m} (1-\alpha)^m \alpha^k \\
&+ \frac{\alpha}{1-\alpha} (1 - \sum_{m=0}^{k-1} \binom{m+k-1}{m} (1-\alpha)^k \alpha^m) \\
&= \frac{\alpha}{1-\alpha} + \sum_{m=0}^{k-1} \binom{m+k-1}{m} \left((1-\alpha)^m \alpha^k - (1-\alpha)^{k-1} \alpha^{m+1} \right)
\end{aligned}
$$

Where in the 6th equation (the one before the last) we used the fact shown in class, that $\Pr[\ell_a = m \mid \ell_h = k] = \binom{m+k-1}{m}(1-\alpha)^k \alpha^m$ and therefore $1 = \sum_{m=0}^{\infty} \Pr[\ell_a = m \mid \ell_h = k] = \sum_{m=0}^{\infty} \binom{m+k-1}{m}(1-\alpha)^k \alpha^m$. □

Question 2. Describe the vector76 attack in terms of the probability of a successful attack. Formally, let $\mathcal{A}$ be the event of a successful double spend attack (with no premining – as you have analyzed in class). Let $\mathcal{B}$ be the event of a successful vector76 attack. Prove that $\mathcal{A} \subseteq \mathcal{B}$ and conclude that the probability of a successful vector76 attack is at least as high as the probability of a successful double-spend attack.

Solution. We want to show that $\mathcal{A} \subseteq \mathcal{B}$, which means that we want to show that if a successful double-spend attack occurred, then an successful vector76 attack occurred as well. From the monotonicity of the probability function, we'll get that the probability of a successful vector76 attack is at least the probability of a successful double-spend attack.

Well, assume that a successful double-spend attack occurred. Let's look at the end of the first phase of the double-spend attack. At this time-step t the honest-nodes had k blocks, and the attacker had m blocks. We separate into two cases - $m > k$ and $m \leq k$.

If $m > k$, the attacker immediately won the "race to defeat" and the double-spend attack was over successfully. In this case, at some previous time-step $t' < t$ the attacker has k blocks,

and the honest-nodes had $k' < k$ blocks. Note that this previous time-step t' is in fact the end of the first phase of the vector76 attack, and the vector76 attack's "race to defeat" was won immediately (because the attacker had k blocks and the honest-nodes has less than k blocks, so he can public his chain to the light-node L and win). Overall we got that if $m > k$ then the vector76 attack was successful, meaning that the event $\mathcal{B}$ occurred.

Now we look at the case $m \leq k$. In this case the "race to defeat" phase of the double-spend attack is not over immediately, and the attacker now needs to accumulate blocks faster than the honest-nodes. Since the double-spend attack was successful, we know that in some later time-step $t' > t$ the attacker had more blocks than the honest-nodes. Assume that in this later time-step t' the attacker has m' blocks, and the honest-nodes had k' blocks, and $m' > k'$. Since $k' \geq k$ (because the honest-nodes started the race with k blocks, and they do not "lose" blocks) we get that the attacker has $m' > k$ blocks. This means that in some time-step t'' which is $t < t'' < t'$ the attacker had k blocks, and the honest-nodes had k'' blocks (such that $k \leq k'' \leq k'$) and the attacker won the vector76 attack's "race to defeat" (which is the same as the regular double-spend attack's "race to defeat"). Overall we got that if $m \leq k$ then the vector76 attack was successful, meaning that the event $\mathcal{B}$ occurred.

To summarize, we proved that if a successful double-spend attack occurred, then a successful vector76 attack occurred, or equivalently $\mathcal{A} \subseteq \mathcal{B}$, as required. $\square$

3 Premining

In this part we will focus on an attacker that cannot choose the time of the attack, but can pre-mine blocks. We will compute the probability that the attacker starts the attack with a certain number of blocks. The double spending attack we analyzed in class assumed the attacker does not start with an advantage of several blocks against the honest network. We later saw that when the attacker creates blocks in advance, he can start the attack with some advantage – he may have some advantage $\mathcal{X}$ of blocks in his secret chain before sending the transaction that will later be double-spent. To do so, the attacker continually tries to mine a secret chain and restarts whenever the honest nodes have a longer chain.

Question 1. Formally describe the pre-mining attempts as a random walk. What are the states of the system? What is the probability of transitioning from state to state?

Solution. In order to describe the pre-mining attempts as a random walk, we'll look at time-steps t which are discrete - time-step t means that the total number of blocks build by the network is t (including the attacker, and the blocks it throws away). Define X_t to be be the random-variable indicating the difference between the length of the attacker's alternative chain, and the length of the honest-nodes' chain. In fact, this is the "advantage" of the attacker in the given time-step t. Note that whenever the the honest-nodes' chain is longer than the attacker's alternative chain, the attacker discards his chain and just wait until he succeeds to mine the next block (on top of the latest block in the honest-nodes' chain) and when he does - he'll has an advantage of 1 blocks (meaning $X_t = 1$). This means that the random-variable X_t never reaches a negative value, it's positive if the attacker has an advantage, and zero otherwise.

So the states of the Markov-chain is the advantage the attacker has in the current time-step, which is $\mathbb{N} \cup \{0\}$. For any $n \geq 1$, the probability of the transition from n to $n-1$ is $1-\alpha$ (because this is the probability that the honest-nodes' will mine the next block). Regarding transition from n to itself - it's 0 for all $n \geq 1$, and $(1-\alpha)$ for $n = 0$ (because if the attacker has no advantage and the honest-nodes' mine another block, his advantage remains zero). Furthermore, for any $n \geq 0$ the probability of the transition from n to $n+1$ is α (this is the probability that the attacker will mine the next block). Note that every other transition that was not mentioned is with probability 0.

Overall we get that for every $n \geq 1$.

$$\Pr[X_t = n+1 \mid X_{t-1} = n] = \alpha$$
$$\Pr[X_t = n-1 \mid X_{t-1} = n] = 1-\alpha$$

and for $n = 0$ we have that

$$\Pr[X_t = 1 \mid X_{t-1} = 0] = \alpha$$
$$\Pr[X_t = 0 \mid X_{t-1} = 0] = 1-\alpha$$

Note that indeed, for every $n \geq 0$ the sum of the transitions from n to other states is exactly 1, as required. □

Question 2. The state space is infinite, so assume that the attacker never acquires more than a 30 block advantage. If the attacker has such an advantage and mines an extra block, that block is discarded.

Write Python 3 code that uses the random walk's transition matrix to compute a good approximation of the stationary distribution. Provide the first 10 probabilities of the result for an attacker that has 25% of the hash-rate. The numpy package can help with matrix operations.

Solution. The Python code is attached, and here are the first 10 probabilities of the result. Clearly, it approaches 0 pretty fast.

$$6.67 \cdot 10^{-1}, 2.22 \cdot 10^{-1}, 7.41 \cdot 10^{-2}, 2.47 \cdot 10^{-2}, 8.23 \cdot 10^{-3},$$
$$2.74 \cdot 10^{-3}, 9.14 \cdot 10^{-4}, 3.05 \cdot 10^{-4}, 1.02 \cdot 10^{-4}, 3.39 \cdot 10^{-5}$$

```python
import numpy as np

MAX_STATES = 30    # The maximal number of states in the Markov chain.
alpha = 0.25       # The amount of compute power the attacker has.

# The maximal allowed difference between the current power of the
# transition matrix and the previous power, as well as the maximal
# allowed difference between any two rows in the final transition
# matrix (i.e. each row approaches the stationary distribution).
epsilon = 0.0001

# We create the transition-matrix of the Markov chain.
# The entry i,j is the probability of the transition from state i to j.
# It's a stochastic matrix, meaning that its entries are non-negative,
# and each row sums to 1 (indeed, for every i the sum over j of the
# probabilities of the transition from i to j is 1).
P = np.zeros(shape=(MAX_STATES, MAX_STATES), dtype=np.float32)
for i in range(MAX_STATES):
    P[i, max(0, i - 1)] = 1 - alpha
    P[i, min(MAX_STATES - 1, i + 1)] = alpha

# Now we raise the matrix P to powers 2^i for i = 1,2,...
# Until we are close to the stationary distribution.
previous_power_matrix = np.eye(len(P))
power_matrix = P
stationary_found = False
while not stationary_found:
    previous_power_matrix = power_matrix
    power_matrix = np.dot(previous_power_matrix, previous_power_matrix)
```

```
    # Check if the two matrices P^{2^i} and P^{2^{i-1}} are the same
    # (up to epsilon).
    matrices_are_close = (
            np.linalg.norm(power_matrix - previous_power_matrix,
                           ord=np.inf) < epsilon
    )

    # (P[i] - P) gets the diff between the i-th row and all other rows.
    # np.delete removes the i-th row from this matrix (since it's zero).
    # np.linalg.norm gets the norm of every row in this diff-matrix.
    # np.amax gets the max distance between the i-th row and another row.
    # Finally, loop over i to find the max distance between any two rows.
    max_diff_between_rows = max(
        np.amax(np.linalg.norm(np.delete(power_matrix[i] - power_matrix, i,
                                         axis=0),
                               axis=0, ord=np.inf))
        for i in range(MAX_STATES)
    )

    # Check that the rows of the matrix are all the same (up to epsilon).
    # Each row approaches the stationary distribution,
    # when the power approaches infinity.
    rows_are_close = (max_diff_between_rows < epsilon)

    # We stop when both the diff between P^{2^{i-1}} and P^{2^i} is small,
    # and the difference between the rows in P^{2^i} is small.
    stationary_found = matrices_are_close and rows_are_close

empirical_stationary_distribution = power_matrix[0]
print(empirical_stationary_distribution)

# Now calculate the theoretical stationary distribution
# and print the maximal difference between it and the empirical.
q = alpha / (1 - alpha)
theoretical_stationary_distribution = ((1 - q) *
                                       np.power(q, np.arange(MAX_STATES)))
print('Maximal diff between empirical and theoretical is {}'.format(
    np.linalg.norm(empirical_stationary_distribution -
                   theoretical_stationary_distribution, ord=np.inf)
))
```

□

Question 3. Prove that the following expression is the stationary distribution:

$$p_n = C\left(\frac{\alpha}{1-\alpha}\right)^n$$

What is the normalization constant C?

Solution. Note that for every $n \geq 1$ it holds that

$$\begin{aligned}\Pr[X_t = n] &= \Pr[X_t = n \mid X_{t-1} = n-1] \Pr[X_{t-1} = n-1] \\ &+ \Pr[X_t = n \mid X_{t-1} = n+1] \Pr[X_{t-1} = n+1]\end{aligned}$$

and for $n = 0$ it holds that

$$\begin{aligned}\Pr[X_t = 0] &= \Pr[X_t = 0 \mid X_{t-1} = 0] \Pr[X_{t-1} = 0] \\ &+ \Pr[X_t = 0 \mid X_{t-1} = 1] \Pr[X_{t-1} = 1]\end{aligned}$$

Note that we already know the transition probabilities, i.e. $\Pr[X_i = m \mid X_{i-1} = m \pm 1]$. The stationary distribution is in fact the probability $\Pr[X_t = n]$ for every n, where t approaches infinity. Well, an important property of the stationary distribution $(p)_{n=1}^{\infty}$, is that if we substitute the terms $\Pr[X_i = m]$ in the above formulas with p_m we'll get an equation. We'll begin with the equation for every $n \geq 1$, and then verify it also works for $n = 0$.

$$\begin{aligned}p_n &= \alpha p_{n-1} + (1-\alpha) p_{n+1} \\ C\left(\frac{\alpha}{1-\alpha}\right)^n &= \alpha C \left(\frac{\alpha}{1-\alpha}\right)^{n-1} + (1-\alpha) C \left(\frac{\alpha}{1-\alpha}\right)^{n+1} \\ C\left(\frac{\alpha}{1-\alpha}\right) &= \alpha C + (1-\alpha) C \left(\frac{\alpha}{1-\alpha}\right)^2 \\ C\left(\frac{\alpha}{1-\alpha}\right) &= \alpha C + C\left(\frac{\alpha^2}{1-\alpha}\right) \\ \alpha C &= \alpha(1-\alpha) C + \alpha^2 C \\ C &= (1-\alpha) C + \alpha C \\ 0 &= (1-\alpha+\alpha-1) C \\ 0 &= 0\end{aligned}$$

we got $0 = 0$ in the end, so the substitution of p_n with the above formula works perfectly.

Now we verify it also works for $n = 0$. Note that if in the previous step the attacker had an advantage of 0 that the probability it remains 0 is $(1-\alpha)$ which is the probability the honest-nodes' will mine the next block. If in the previous step the attacker had an advantage of 1 that the probability it becomes 0 is also $(1-\alpha)$ which is the probability the honest-nodes' will mine the next block. Overall we get

$$\begin{aligned}p_0 &= (1-\alpha) p_0 + (1-\alpha) p_1 \\ \alpha p_0 &= (1-\alpha) p_1 \\ \alpha C \left(\frac{\alpha}{1-\alpha}\right)^0 &= (1-\alpha) C \left(\frac{\alpha}{1-\alpha}\right)^1 \\ \alpha C &= \alpha C \\ 0 &= 0\end{aligned}$$

and again we got $0 = 0$ so overall the substitution of $(p_n)_{n=1}^{\infty}$ with the formulas works perfectly, and we proved that $(p_n)_{n=1}^{\infty}$ is indeed the stationary distribution.

In order to calculate the normalization factor C, we need to take into account the fact that this is a probability vector, which means that is sums to 1. Well, note that p_n is an infinite geometric series, with initial value C and quotient $\frac{\alpha}{1-\alpha}$ so we can use the formula for the sum of infinite geometric series to get

$$1 = \sum_{n=0}^{\infty} p_n = \sum_{n=0}^{\infty} C \left(\frac{\alpha}{1-\alpha}\right)^n = \frac{C}{1-\frac{\alpha}{1-\alpha}} = \frac{C}{1-\frac{\alpha}{1-\alpha}}$$

So we get that

$$C = 1 - \frac{\alpha}{1-\alpha}$$

To summarize, we proved that the stationary distribution is

$$\forall n \geq 0, p_n = (1 - \frac{\alpha}{1-\alpha}) \left(\frac{\alpha}{1-\alpha}\right)^n$$

as required. □

Question 4. Show that the analytical result matches the stationary distribution you computed for an attacker with 25% of the compute power.

Solution. By using the Python code mentioned previously, we get that the maximal ℓ_∞-distance between the empirical stationary distribution and the theoretical one is 0.000000576, which means that the theoretical and empirical stationary distributions indeed match (of course we could have computed a larger power of the transition matrix to get a smaller difference). □