NYILATKOZAT

Név: Kovács Botond

ELTE Természettudományi Kar, szak: Matematika BSc

NEPTUN azonosító: LG308Z

Szakdolgozat címe:
Markov-láncok és alkalmazásaik eszközárazási feladatokban

A **szakdolgozat** szerzőjeként fegyelmi felelősségem tudatában kijelentem, hogy a dolgozatom önálló szellemi alkotásom, abban a hivatkozások és idézések standard szabályait következetesen alkalmaztam, mások által írt részeket a megfelelő idézés nélkül nem használtam fel.

Budapest, 2022. május 20.

Kovács Botond

a hallgató aláírása

Markov-láncok és alkalmazásaik eszközárazási feladatokban

Szakdolgozat

Készítette: Kovács Botond
Matematika BSc

Témavezető: Dr. Csiszár Villő
Valószínűségelméleti és Statisztikai Tanszék

Eötvös Loránd Tudományegyetem
Természettudományi Kar
2022
Budapest

Tartalomjegyzék

Ábrák jegyzéke

Köszönetnyilvánítás

Hatalmas köszönettel tartozom témavezetőmnek, Dr. Csiszár Villőnek a témaajánlásért, a konzultációkért és a számtalan emailért és Teams üzenetért, amik elősegítették szakdolgozatom elkészülését.

Szeretném megköszönni családomnak, barátnőmnek és barátaimnak, hogy ilyen vagy olyan módon hozzájárultak a tanulmányaimban való előrehaladásomhoz és szakdolgozatom megírásához.

1. Bevezetés

Szakdolgozatom egy rendkívül leegyszerűsített világban játszódik. Mondhatni, hogy ebben a világban mindenki a mának él, mivel senkit sem izgat az, hogy tegnap, vagy azelőtt mi történt. Semmilyen hatása nincsen a holnapra, az csakis a mai napnak a függvénye. Nem tudom, hogy ez egy ideálisabb világ-e, könnyebb lehet-e így boldogulni, ezt mindenki döntse el maga. Nem is ezt a kérdéskört boncolgatom, hanem azt, hogy egy ilyen univerzumban lévő idősorra milyen szabályok vonatkoznak.

Egész pontosan a Markov-láncok témakörébe nyertem betekintést dolgozatom megírása során, különös tekintettel arra, hogy milyen segítséget nyújthatnak nekünk egy pénzügyi eszköz árazásában. A Markov-lánc az egyik legegyszerűbb matematikai modell, lényegében azután következik a sorban, hogy minden esemény egymástól független.

Szakdolgozatom elkészítése közben sokat programoztam Python programozási nyelven, ugyanis egyik további fő eleme a Python egy Quantecon nevezetű könyvtárában található beépített függvények ismertetése. A QuantEcon projekt célja egy modern, nyílt kódú közgazdaságtani eszköstár fejlesztése.[1]

Ennek megfelelően a 2. fejezetben a legfontosabb definícióit és tulajdonságait foglalom össze a Markov-láncoknak. Ezzel párhuzamosan mutatom be a beépített függvények használatát is.

A 3. fejezetben az egyik fontos alkalmazásukról van szó, az autoregresszív folyamatok közelítéséről. Két módszert prezentálok részletesen, melyek más-más módon közelítenek egy ilyen idősort, de mindkét esetben Markov-láncokat használunk. Ennek érdekében először az AR(1) folyamat legfontosabb tudnivalóit ismertetem, és kiszámolom többek között a várható értékét, szórásnégyzetét. Erre azért lesz szükség, mivel a második metódus az említett jellemzőkből kiindulva épít fel egy Markov-láncot, míg az első módszer az átmenetvalószínűségeket diszkretizálja. Közben látható az is, hogyan implementálom a két módszert Python segítségével.

A 4. fejezetben egy leegyszerűsített, véletlent használó matematikai modellt építek fel, melynek motivációjaként bemutatásra kerül két másik is. Mindhárom esetben kiszámolom egy osztalékot termelő pénzügyi eszköz egyensúlyi árát. Ebben a fejezetben is programkódokkal egészítem ki a tartalmat, az egyensúlyi árak számítása és a különböző idősorok vizualizációja is látható lesz.

[1] A projekt honlapja: https://quantecon.org/

2. Markov-láncok

Ebben a fejezetben a sztochasztikus folyamatok egy speciális osztályát, a Markov-láncokat ismertetem az [1] és [5] források alapján, illetve ezzel párhuzamosan bemutatom a QuantEcon.py könyvtár Markov-láncokkal kapcsolatos alapvető parancsait, melyeket a későbbiekben alkalmazni fogok. Ehhez elengedhetetlen a könyvtár installálása és a használandó kiegészítő csomagok importálása:

```
!conda install -y quantecon

%matplotlib inline
import math
import numpy as np
import matplotlib.pyplot as plt
import quantecon as qe
import pandas as pd
import statistics as stat
from scipy.stats import norm
from numpy.linalg import eigvals, solve
```

2.1. Markov-tulajdonság

Sztochasztikus folyamat alatt lényegében valószínűségi változók egy indexelt családját értjük. Formálisan legyen $(\Omega, \mathcal{F}, \mathbb{P})$ valószínűségi mező, $(I, \mathcal{B})$ mérhető tér és T vagy a természetes számok halmaza, vagy a nemnegatív valós számok halmaza. Sztochasztikus folyamatnak nevezzük az $X_t : \Omega \to I \, (t \in T)$ valószínűségi változókból álló, idő szerint indexelt halmazt. A Markov-tulajdonságú folyamat, vagy röviden Markov-folyamat olyan folyamat, amelyben a jövőbeli állapotok nem függnek a múltbeli állapotoktól, csak a jelenbélitől.

2.1.1. Definíció. *Minden $t \in T$ időpontra legyen $\mathcal{F}_t := \sigma(X_s : s \leq t)$. Egy sztochasztikus folyamat Markov-tulajdonságú, ha minden $B \in \mathcal{B}$ halmazra és minden $t > s$ esetén*

$$\mathbb{P}(X_t \in B \,|\, \mathcal{F}_s) = \mathbb{P}(X_t \in B \,|\, X_s).$$

A Markov-folyamatot Markov-láncnak hívjuk, ha az I halmaz megszámlálható. Ekkor őt a Markov-lánc állapotterének, elemeit állapotértékeknek vagy állapotoknak hívjuk. Könnyedén észrevehető, hogy így egyrészt $\mathcal{B}$ az I összes részhalmazából áll, másrészt az Ω eseménytér felbomlik megszámlálható sok atomra, amik generálják $\mathcal{F}_n$-et, ezért a definícióban szereplő egyenlet úgy írható át, hogy minden $m \geq n$, $i_0, i_1, ..., i_n, i_m \in I$ esetén

$$\mathbb{P}(X_m = i_m \,|\, X_0 = i_0, X_1 = i_1, ..., X_n = i_n) = \mathbb{P}(X_m = i_m \,|\, X_n = i_n).$$

A könnyebb érthetőség érdekében vegyünk példának egy tetszőleges gráfot, aminek a csúcsain bolyongunk úgy, hogy egy adott csúcsból a szomszédos csúcsokba tudunk továbblépni egyenlő valószínűséggel. X_t jelölje azt, hogy a t. lépésben melyik csúcsban vagyunk. Ekkor az $\{X_t\}_{t=0,1,\dots}$ sorozat egy Markov-lánc, hiszen egy ismert múltbéli sétának csak az utolsó csúcsától függ az, hogy melyik csúcsba lépünk egy későbbi időpontban.

2.2. Markov-láncok karakterizációja

Minden Markov-láncot az átmeneti mátrixa és a kezdeti eloszlása határoz meg. A Markov-láncok egy további fontos tulajdonsága, mely a definícióból nem következik, de mindig fel szoktuk tenni, hogy az átmenetvalószínűségek stacionáriusak, másnéven homogének, ami azt jelenti, hogy

$$\mathbb{P}(X_m = j \,|\, X_n = i) = p_{ij}^{(m-n)}.$$

Vagyis az egyik állapotból a másikba jutás csak a közöttük eltelt idő függvénye, az nem számít, hogy addig mennyi idő telt el. Az egyszerűség kedvéért jelöljük a $\mathbb{P}(X_{n+1} = j \,|\, X_n = i)$ valószínűségeket $p_{ij}^{(1)}$ helyett p_{ij}-vel, és nevezzük egylépéses átmenetnek. A $P = (p_{ij})_{i,j\in I}$ mátrixot átmenetmátrixnak nevezzük.

2.2.1. Példa. *Tegyük fel, hogy egy munkát kereső ember minden t időpillanatban vagy munkanélküli, vagy munkavállaló, és teljesülnek az alábbiak:*

- *egy munkanélküli α valószínűséggel talál munkát*
- *egy munkavállaló β valószínűséggel veszíti el a munkáját*
- $\alpha, \beta \in (0, 1)$

Ekkor az átmenetvalószínűségekkel elkészíthető az alábbi mátrix:

$$P = \begin{pmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{pmatrix}.$$

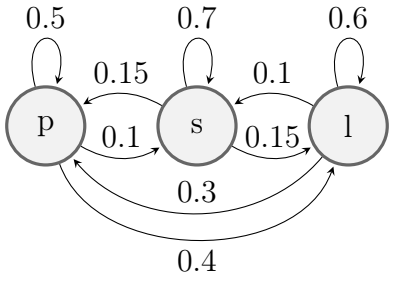
Egy ilyen modellben, ha ismerjük α-t és β-t, meg tudunk válaszolni olyan kérdéseket, hogy mekkora valószínűséggel veszíti el valaki a munkáját egy év alatt, átlagosan mennyi ideig marad munkanélküli egy aktív munkakereső, vagy hogy hosszú távon mi az aránya a munkanélkülieknek és a munkavállalóknak.

Amennyiben az állapottér véges, az állapotokat és az átmenetvalószínűségeket tudjuk ábrázolni gráfként is. A gráf csúcsai felelnek meg az állapotoknak, az élei pedig az átmenetvalószínűségeknek.

2.2.2. Példa. *Legyen adott egy étterem, amelynek az a különlegessége, hogy minden héten csak egyféle ételt szolgálnak fel az alábbiak közül: pizzák, szendvicsek, lángosok. Tehát az állapottér* $\{p, s, l\}$*, a változtatásokat jellemző valószínűségeket pedig tartalmazza az alábbi mátrix:*

$$P = \begin{pmatrix} 0.5 & 0.1 & 0.4 \\ 0.15 & 0.7 & 0.15 \\ 0.3 & 0.1 & 0.6 \end{pmatrix}.$$

Például ha ezen a héten pizzát szolgálnak fel, akkor annak a valószínűsége, hogy jövő héten lángost fognak 0.4. A Markov-lánc gráfként ábrázolva:



1. ábra. Egy Markov-lánc gráfja

2.2.1. Tétel. *Jelöljük* $p_{ij}^{(n)}$*-nel az n-lépéses átmenetvalószínűségeket, tehát* $p_{ij}^{(n)} = \mathbb{P}(X_{m+n} = j \,|\, X_m = i)$*. Ezekre teljesül a Chapman-Kolmogorov egyenlőség:*

$$p_{ij}^{(l+m)} = \sum_{k \in I} p_{ik}^{(l)} p_{kj}^{(m)}.$$

Bizonyítás:

$$p_{ij}^{(l+m)} = \mathbb{P}(X_{l+m} = j \,|\, X_0 = i) = \sum_k \mathbb{P}(X_{l+m} = j, X_l = k \,|\, X_0 = i) =$$

$$= \sum_k \mathbb{P}(X_l = k \,|\, X_0 = i)\mathbb{P}(X_{l+m} = j \,|\, X_l = k, X_0 = i) = \sum_{k \in I} p_{ik}^{(l)} p_{kj}^{(m)}.$$

■

Ez azt jelenti, hogy a $p_{ij}^{(n)}$ mennyiségek a P^n mátrix megfelelő elemei.

Ha QuantEcon.py-t használva szeretnénk létrehozni egy Markov-láncot, akkor elegendő megadni az átmenetmátrixát, de megadhatóak még az állapotok nevei is:

```
P = np.array([[0.5, 0.1, 0.4],
              [0.15, 0.7, 0.15],
              [0.3, 0.1, 0.6]])

mc = qe.MarkovChain(P, state_values=('p', 's', 'l'))
mc.simulate(10)
```

```
array(['p', 's', 's', 'p', 'l', 'l', 'p', 'p', 'p', 'p'], dtype='<U1')
```

Itt a 2.2.2 Példában szereplő Markov-láncból generálunk egy 10 hosszúságú blokkot, az állapotok p (pizza), s (szendvics) és l (lángos).

Ahhoz, hogy egy Markov-lánc egyértelmű legyen, tudni kell még a kiindulási állapotot, vagy annak az eloszlását.

Egy Markov-lánc kezdeti eloszlásán az X_0 valószínűségi változó eloszlását értjük, és μ-vel jelöljük. Az átmenetmátrix és a kezdeti eloszlás együtt már egyértelműen meghatározzák a Markov-láncot, mivel ekkor ki tudjuk számolni a véges dimenziós eloszlásokat:

$$\mathbb{P}(X_0 = i_0, X_1 = i_1, ..., X_n = i_n) =$$

$$= \mathbb{P}(X_0 = i_0)\mathbb{P}(X_1 = i_1 \,|\, X_0 = i_0)\cdot ... \cdot \mathbb{P}(X_n = i_n \,|\, X_0 = i_0, X_1 = i_1, ..., X_{n-1} = i_{n-1}) =$$

$$= \mu(i_0)p_{i_0 i_1} \cdot ... \cdot p_{i_{n-1} i_n}.$$

Ezek alapján a Markov-lánc akárhányadik elemének eloszlását meg tudjuk adni. Jelöljük X_t eloszlását ψ_t-vel. Tehát $\psi_0 = \mu$. A teljes valószínűség tételét és a Markov-tulajdonságot használva:

$$\mathbb{P}(X_{t+1} = j) = \sum_{i \in I} \mathbb{P}(X_{t+1} = j \,|\, X_t = i)\mathbb{P}(X_t = i).$$

Továbbalakítva:

$$\psi_{t+1}(j) = \sum_{i \in I} p_{ij}\psi_t(i).$$

Ha ψ-re úgy gondolunk, mint egy sorvektorra, akkor az előbbi egyenleteket összegezve kapjuk:

$$\psi_{t+1} = \psi_t P.$$

Ez azt jelenti, hogy ha egy lépéssel későbbi eloszlást szeretnénk megkapni, akkor az eddigi eloszlást meg kell szoroznunk a P átmenetmátrixszal. Ezt iterálva $\psi_{t+m} = \psi_t P^m$ is helytálló, ezért ha a kezdeti eloszlás μ, akkor X_m eloszlása μP^m.

A MarkovChain osztályban direkt módon nem adható meg a kezdeti eloszlás, az alapértelmezetten egyenletes eloszlás az állapotok között. Viszont szimuláláskor megadható a kezdeti állapot, amihez a NumPy random.choice függvényét használva beállítható rá tetszőleges eloszlás, ami így lényegében a kezdeti eloszlás lesz:

```
P = np.array([[0.5, 0.1, 0.4],
              [0.15, 0.7, 0.15],
              [0.3, 0.1, 0.6]])

μ = [0.2, 0.35, 0.45]

mc = qe.MarkovChain(P)
mc.simulate(10, init = np.random.choice([0, 1, 2], p = μ))
```

```
array([2, 2, 0, 1, 0, 2, 2, 0, 0, 2])
```

Az éttermes példánál maradva létrehozunk egy μ eloszlásvektort, majd a szimulálásnál az *init* paraméterként megadva érhető el, hogy ez legyen a Markov-lánc kezdeti eloszlása.

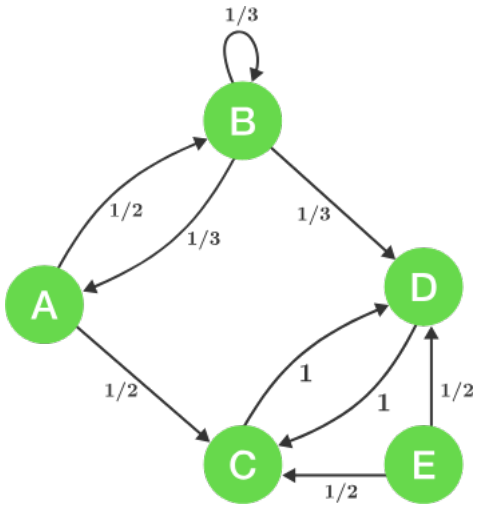
2.3. Markov-lánc állapotainak osztályozása

Érdekes kérdés a Markov-láncokkal kapcsolatban, hogy melyik állapotból melyikbe lehet eljutni, van-e olyan, amiből nem lehet kijutni, vagy hogy milyen gyakran térünk vissza egy adott állapotba. Ebben az alfejezetben ezeket a kérdéseket boncolgatjuk.

2.3.1. Definíció. *Azt mondjuk, hogy az i állapotból elérhető j ($i \to j$), ha $\exists n \geq 0$, amire $p_{ij}^{(n)} > 0$. Az i és j állapotok közlekednek, ha $i \to j$ és $j \to i$. Ez ekvivalenciareláció, tehát osztályokra bontja az állapotteret. Egy Markov-lánc irreducibilis, ha egy osztályból áll.*

Ezen felül az i állapotot lényegesnek mondjuk, ha $i \to j$-ből következik, hogy $j \to i$. Az i állapot lényegtelen, ha nem lényeges. Triviális, hogy a lényegesség is osztálytulajdonság, vagyis egy osztályon belül vagy minden állapot lényeges, vagy egyik sem (vagyis minden állapot lényegtelen). A lényeges osztályokból nem lehet kijutni, hiszen akkor vissza is tudnánk, de ez csak osztályon belül történhet, a lényegtelen osztályokból pedig ki lehet jutni, viszont vissza már nem. Továbbá az i állapotot elnyelő állapotnak hívjuk, ha $p_{ii} = 1$.

Viszgáljuk meg a következő ábrán látható Markov-láncot az osztályai szempontjából. Az A és B állapotokba egymásból lehet eljutni, viszont ezen felül csak kijutni tudunk, tehát ők egy lényegtelen osztályt alkotnak. Ugyanez igaz az E állapotra, belőle csak kijutni lehet. A harmadik osztályt a C és D állapotok alkotják, belőlük máshova nem lehet eljutni, tehát ők egy lényeges osztály. Ez azt is jelenti, hogy ez a Markov-lánc nem irreducibilis, hiszen 3 osztály alkotja.



2. ábra. Markov-lánc 3 osztállyal
https://brilliant.org/practice/stability/

Ellenőrizzük le magunkat a beépített függvények segítségével:

```
P = np.array([[0, 1/2, 1/2, 0, 0],
              [1/3, 1/3, 0, 1/3, 0],
              [0, 0, 0, 1, 0],
              [0, 0, 1, 0, 0],
              [0, 0, 1/2, 1/2, 0]])

mc = qe.MarkovChain(P,['A','B','C','D','E'])
mc.communication_classes
```

```
[array(['C', 'D'], dtype='<U1'),
 array(['A', 'B'], dtype='<U1'),
 array(['E'], dtype='<U1')]
```

```
mc.is_irreducible
```

```
False
```

Egy állapot periódusán azt értjük, hogy hány lépésenként tudunk oda visszatérni. Ha nincsen szabályos ismétlődés a visszatérések számában, akkor aperiodikusnak fogjuk hívni.

2.3.2. Definíció. *Az i állapot periódusa a $\{n > 0 : p_{ii}^{(n)} > 0\}$ halmaz legnagyobb közös osztója, $d(i)$-vel jelöljük. Amennyiben $d(i) = 1$, akkor az i állapot aperiodikus, ha pedig a halmaz üres, akkor nem értelmezzük az periódusát.*

2.3.1. Állítás. *Egy osztály minden állapotának ugyanannyi a periódusa.*

Bizonyítás: Legyenek i és j azonos osztálybeli különböző állapotok. Ekkor biztosan értelmezzük $d(i)$-t és $d(j)$-t. Tudjuk, hogy $\exists\, n, m$, amikre $p_{ij}^{(n)} > 0$ és $p_{ji}^{(m)} > 0$. Legyen s olyan, hogy $p_{jj}^{(s)} > 0$. Ekkor teljesül $p_{ii}^{(n+m)} > 0$ és $p_{ii}^{(n+s+m)} > 0$, vagyis $d(i)|n+m$ és $d(i)|n+s+m$, amiből $d(i)|s$. Tehát $d(i)$ közös osztója az ilyen s számoknak, $d(j)$ pedig definíciójából adódva a legnagyobb közös osztója ezen számoknak. Ezért $d(i)|d(j)$, és mivel i és j szerepe felcserélhető, ezért $d(i) = d(j)$. ■

Az előző példát vizsgálva azt látjuk, hogy az $\{A, B\}$ osztály aperiodikus, hiszen $p_{BB} = 1/3$. Az $\{E\}$ osztály periódusát nem értelmezzük, mivel nem lehet bele visszatérni, a $\{C, D\}$ osztály periódusa pedig 2, mivel minden második lépésben visszatérünk az adott csúcsba. A QuantEcon-ban rendelkezésünkre áll olyan függvény, amely megadja a periódust, viszont csak abban az esetben, ha a Markov-lánc irreducibilis, ezért a 2.2.2 Példán mutatom be:

```
P = np.array([[0.5, 0.1, 0.4],
              [0.15, 0.7, 0.15],
              [0.3, 0.1, 0.6]])
mc = qe.MarkovChain(P)
mc.period
```

```
1
```

```
mc.is_aperiodic
```

```
True
```

Hasznos még megismerkednünk a következő jelölésekkel is:

$$f_{ij}^{(0)} = 0,\ f_{ij}^{(n)} = \mathbb{P}(X_n = j,\, X_k \neq j : k = 1, 2, ..., n-1 \,|\, X_0 = i)\ n \geq 1,$$

$$f_{ij}^{*} = \sum_{n=1}^{\infty} f_{ij}^{(n)}.$$

Vagyis $f_{ij}^{(n)}$ annak a valószínűsége, hogy az i állapotból indulva először az n. lépésben ér el a j állapotba a lánc, f_{ij}^{*} pedig annak a valószínűsége, hogy az i állapotból valamikor elérünk a j állapotba.

2.3.3. Definíció. *Egy i állapotot rekurrensnek hívunk, ha $f_{ii}^* = 1$, és tranziensnek, ha nem rekurrens. Ha rekurrens, akkor értelmezzük az átlagos visszatérési időt, ami $m_i = \sum_{n=1}^{\infty} n f_{ii}^{(n)}$. Az i állapot pozitív rekurrens, ha $m_i < \infty$, és nulla rekurrens, ha $m_i = \infty$. A tranziens, nulla rekurrens és pozitív rekurrens tulajdonságok osztálytulajdonságok.*

2.3.2. Állítás. *Legyen $\{X_n\}_{n\geq 0}$ véges állapotterű Markov-lánc. Ekkor (i) minden rekurrens osztály pozitív, valamint (ii) minden tranziens osztály lényegtelen, és a lánc 1 valószínűséggel előbb-utóbb elhagyja.*

Bizonyítás:

(i) Legyen C egy rekurrens osztály állapotainak halmaza. Ekkor minden $i \in C$-re és $n \geq 1$-re

$$\sum_{j\in C} p_{ij}^{(n)} \;=\; 1.$$

Belátható, hogy ha j tranziens vagy nulla rekurrens állapot, akkor $\lim_{n\to\infty} p_{ij}^{(n)} = 0$. Ebből adódóan ha C nulla rekurrens lenne, akkor teljesülnie kellene, hogy minden $j \in C$-re $\lim_{n\to\infty} p_{ij}^{(n)} = 0$, ami most ellentmondás, mert C véges.

(ii) Legyen C egy tranziens osztály, ekkor minden $i \in C$-re és $n \geq 1$-re

$$1 = \sum_{j\in C} p_{ij}^{(n)} + \sum_{j\notin C} p_{ij}^{(n)}.$$

Mindkét oldal határértékét véve:

$$1 = \sum_{j\in C} \lim_{n\to\infty} p_{ij}^{(n)} + \lim_{n\to\infty} \sum_{j\notin C} p_{ij}^{(n)} = \sum_{j\in C} 0 + \lim_{n\to\infty} \mathbb{P}(X_n \notin C \,|\, X_0 = i) =$$
$$= \mathbb{P}(\exists\, n : X_n \notin C \,|\, X_0 = i).$$

Tehát valóban 1 a valószínűsége annak, hogy a lánc elhagy egy C tranziens osztályt. ■

Ez alapján minden véges állapotterű Markov-láncnak van legalább egy pozitív osztálya, tehát ha irreducibilis, akkor pozitív rekurrens.

2.4. Stacionárius eloszlás

Ebben a fejezetben azt vizsgáljuk, hogy adható-e meg olyan kezdeti eloszlás, hogy az időben előrehaladva ne változzon.

2.4.1. Definíció. *A μ kezdeti eloszlás stacionárius, ha $\mu = \mu P$. Ha ez az egyenlet teljesül, akkor minden t-re teljesülni fog, hogy $\mu = \mu P^t$.*

Véges állapotterű Markov-láncokra teljesül az alábbi két tétel:

2.4.1. Tétel. *Minden P átmeneti mátrixhoz létezik legalább egy stacionárius kezdeti eloszlás.*

2.4.2. Tétel. *Amennyiben az $\{X_t\}_{t\in T}$ Markov-lánc irreducibilis és aperiodikus, a P átmenetmátrixához pontosan egy μ^* stacionárius kezdeti eloszlása van, és akármilyen μ kezdeti eloszlásra igaz lesz, hogy $\lim_{t\to\infty} ||\mu P^t - \mu^*|| \to 0$.*

Az például elégséges feltétel, hogy az átmenetmátrix minden eleme legyen pozitív, mivel ekkor irreducibilis és aperiodikus lesz a hozzá tartozó Markov-lánc.

A 2.2.2 példán mutatom be a *stationary_distributions* függvényt, ami az összes stacionárius eloszlást megadja.

```
P = np.array([[0.5, 0.1, 0.4],
              [0.15, 0.7, 0.15],
              [0.3, 0.1, 0.6]])

mc = qe.MarkovChain(P)
mc.stationary_distributions
```

```
array([[0.328125, 0.25    , 0.421875]])
```

```
μ_1 = np.array([0.25, 0.25, 0.5])
np.matmul(μ_1,np.linalg.matrix_power(P, 10))
```

```
array([0.32812499, 0.25      , 0.42187501])
```

```
μ_2 = np.array([1/3, 1/3, 1/3])
np.matmul(μ_2,np.linalg.matrix_power(P, 30))
```

```
array([0.32812499, 0.25000002, 0.42187499])
```

Az első output a Markov-lánc stacionárius eloszlása. Utána azt láthatjuk, hogy tetszőleges kezdeti eloszlásokat a P átmenetmátrix hatványával szorozva közelítünk a stacionárius eloszláshoz.

Természetesen nem csak véges állapotterű Markov-láncoknak létezhet stacionárius eloszlása. Nézzünk egy példát végtelen állapotterűre:

2.4.1. Példa. *X_n jelölje azt, hogy egy érmedobás sorozat végén hány fej van, ha az érmét dobálva a fej valószínűsége p. Ekkor X_n egy irreducibilis, aperiodikus Markov-lánc, melynek állapothalmaza a természetes számok. Ha nem az első dobástól kezdjük X_n-t, hanem azt mondjuk, hogy már adva van előttünk egy dobássorozat, akkor értelmes feltenni azt a kérdést, hogy mi a stacionárius eloszlás. Ehhez ki kell számolnunk a $\mu = \mu P$ egyenletrendszert:*

$$\mu(0) = (1-p)\sum_{k=0}^{\infty}\mu(k) = (1-p)\cdot 1 = 1-p,\ \mu(i) = p\mu(i-1) = p^i(1-p),$$

mivel $\mu(0) = 1-p$. Tehát a stacionárius kezdeti eloszlás az $1-p$ paraméterű geometriai eloszlás azon változata, amikor az első siker előtt pontosan i sikertelen kísérletnek kell lennie, vagyis μ olyan, hogy $\mu(i) = \mathbb{P}(X = i) = (1-p)p^i$.

3. AR(1) folyamatok közelítése Markov-lánccal

A későbbiekben használandó egyik modell fontos eleme, hogy a világgazdaság állapotát szimulálni szeretnénk. Az előző fejezet elolvasása után természetes ötletnek hat, hogy használjunk egy Markov-láncot, melynek az állapottere a világ lehetséges állapotainak halmaza, az átmenetvalószínűségek pedig értelemszerűen az egyik állapotból a másikba kerülés valószínűségei.

Azonban egy Markov-lánc megadásához rengeteg paraméter kell. Az állapotok száma (n), az n^2 átmenetvalószínűség és a kezdeti eloszlás vektora (μ). Ezért egy másik módszerhez nyúlunk, az elsőrendű autoregresszióhoz, mely egy olyan alapvető modell, ami általában jól közelíti a gyakorlatban előforduló folyamatok egy részét.

Ennek a sztochasztikus folyamatnak, vagy másnéven idősornak folytonos az állapottere, ami miatt nehezen szimulálható. Itt mégicsak előjönnek a véges állapotterű Markov-láncok, mivel velük fogjuk közelíteni az elsőrendű autoregressziós folyamatot. Ennek hatására az első gondolathoz képest jóval leegyszerűsödik a problémánk, hiszen látni fogjuk, hogy az elsőrendű autoregressziónak csupán 2 paramétere van, a közelítéséhez pedig legfeljebb további 2-re lesz szükségünk.

Tehát a következőkben ismertetem, hogy az idősorok egy speciális halmazát, az elsőrendű autoregressziót hogyan tudjuk kétféleképpen közelíteni véges állapotterű Markov-lánccal. A fejezetet a [2], [3] és [4] forrásokat feldolgozva készítettem.

3.1. Elsőrendű autoregresszió

3.1.1. Elméleti alapok

Először is szükségünk lesz néhány definícióra az idősorokkal kapcsolatban, köztük az AR(1) modellre.

3.1.1. Definíció. *Az $\{\varepsilon_t\}_{t\in T}$ folyamat fehér zaj, ha ε_t-k korrelálatlan, azonos eloszlású valószínűségi változók 0 várható értékkel. Amennyiben függetlenek is, akkor $\{\varepsilon_t\}_{t\in T}$ független értékű zaj.*

3.1.2. Definíció. *Az egyszerűség kedvéért a továbbiakban legyen $T = \mathbb{N}$. Az elsőrendű autoregressziós folyamat, röviden AR(1), egy olyan idősor, amely minden t-re kielégíti az*

$$X_t = \alpha X_{t-1} + \varepsilon_t$$

egyenletet, ahol az X_0 valószínűségi változó előre adott, ε_t független értékű zaj, $\alpha \in \mathbb{R}$.

Tehát az autoregressziós folyamat változók olyan sorozata, melyben egy adott időpillanatban a változó értéke csak az előző pillanatbeli értékétől, és a véletlentől függ.

A folyamat állapottere legtöbbször folytonos, többek között azért, mivel az ε_t általában normális eloszlású. Ilyenkor az AR(1) folyamatot Gauss-AR(1) folyamatnak hívjuk.

3.1.3. Definíció. *Az $\{X_t\}_{t\in\mathbb{N}}$ idősor gyengén stacionárius, ha*

- $\mathbb{E}[X_t] = \mathbb{E}[X_0] \,\forall\, t$*-re,*
- $\mathrm{Cov}[X_t, X_s] = R(t-s) \;\forall\; t \geq s$*-re,*

vagyis a kovarianca csak az eltelt időtől függ.

3.1.4. Definíció. *Az $\{X_t\}_{t\in\mathbb{N}}$ idősor erősen stacionárius, ha véges dimenziós eloszlásai eltolásinvariánsak, vagyis $\forall\, n$-re és $t_1, ..., t_n$-re $(X_{t_1}, ..., X_{t_n}) \sim (X_{t_1+h}, ..., X_{t_n+h})$. Triviális, hogy egy erősen stacionárius idősor gyengén stacionárius is.*

3.1.5. Definíció. *Az $\{X_t\}_{t\in\mathbb{N}}$ idősor oksági megoldása az AR(1) egyenletének, ha a folyamat jelene független a zaj jövőjétől, vagyis minden t-re X_t és $(\varepsilon_{t+1}, \varepsilon_{t+2}, ...)$ függetlenek.*

3.1.2. Momentumok kiszámítása

A továbbiakban bemutatandó egyik módszerhez elengedhetetlen, hogy megadjuk egy bizonyos tulajdonságokkal rendelkező AR(1) folyamat első két momentumát.

3.1.1. Tétel. *Ha adott egy $X_t = \alpha X_{t-1} + \varepsilon_t$ autoregressziós egyenlet, ahol $|\alpha| < 1$, akkor létezik erősen stacionárius oksági megoldása.*

Várható érték:

Amennyiben a fenti feltételek teljesülnek, és $\{X_t\}_{t\in\mathbb{N}}$ erősen stacionárius, akkor a várható érték minden t-re ugyanannyi. Ezt jelöljük μ-vel, és számoljuk ki:

$$\mathbb{E}[X_t] = \mathbb{E}[\alpha X_{t-1} + \varepsilon_t]$$

$$\mathbb{E}[X_t] = \alpha\mathbb{E}[X_{t-1}] + \mathbb{E}[\varepsilon_t]$$

$$\mu = \alpha\mu$$

Amiből $\mu = 0$, mivel $|\alpha| < 1$.

Szórásnégyzet:

Mivel feltettük, hogy X_t oksági megoldás, ezért a szórásnégyzetet is ki tudjuk számolni. Jelölése legyen σ_X^2, ε_t szórásnégyzete pedig σ_ε^2, ekkor:

$$\mathbb{D}^2[X_t] = \mathbb{D}^2[\alpha X_{t-1} + \varepsilon_t]$$

$$\mathbb{D}^2[X_t] = \alpha^2 \mathbb{D}^2[X_{t-1}] + \mathbb{D}^2[\varepsilon_t] + 2\mathrm{Cov}[\alpha X_{t-1}, \varepsilon_t]$$

$$\sigma_X^2 = \alpha^2 \sigma_X^2 + \sigma_\varepsilon^2 + 0$$

$$\sigma_X^2 = \frac{\sigma_\varepsilon^2}{1-\alpha^2}.$$

Feltételes várható érték:

Adott egy tetszőleges $x \in \mathbb{R}$.

$$\mathbb{E}[X_{t+1} \,|\, X_t = x] = \mathbb{E}[\alpha X_t + \varepsilon_{t+1} \,|\, X_t = x] = \mathbb{E}[\alpha x + \varepsilon_{t+1}] = \alpha x.$$

Feltételes szórásnégyzet:

Ismét legyen adott egy tetszőleges $x \in \mathbb{R}$.

$$\begin{aligned}\mathbb{D}^2[X_{t+1} \,|\, X_t = x] &= \mathbb{D}^2[\alpha X_t + \varepsilon_{t+1} \,|\, X_t = x] = \\ &= \mathbb{D}^2[\alpha x + \varepsilon_{t+1}] = \mathbb{D}^2[\varepsilon_{t+1}] = \sigma_\varepsilon^2.\end{aligned}$$

Autokovariancia és autokorreláció:

A definícióban láttuk, hogy stacionárius idősor esetén beszélhetünk autokorrelációs függvényről, ilyenkor ez egy egyváltozós függvény. Továbbra is feltesszük, hogy a megoldás oksági, ekkor

$$\begin{aligned}R(\tau) = \mathrm{Cov}[X_t, X_{t+\tau}] &= \mathrm{Cov}[X_t, \alpha X_{t+\tau-1} + \varepsilon_{t+\tau}] = \\ &= \alpha \cdot \mathrm{Cov}[X_t, X_{t+\tau-1}] = \alpha \cdot R(\tau - 1).\end{aligned}$$

Mivel $R(0) = \mathbb{D}^2[X_t] = \sigma_X^2$, ezért $R(\tau) = \sigma_X^2 \cdot \alpha^\tau = \frac{\alpha^\tau}{1-\alpha^2} \cdot \sigma_\varepsilon^2$. Továbbá ha $r(\tau)$-val jelöljük az autokorrelációs függvényt, akkor $r(\tau) = \frac{R(\tau)}{R(0)} = \alpha^\tau$.

A momentumok kiszámítása után ismertetem az említett két metódust, melyeket később szimulációk során használni is fogok. Az első a szakirodalomban leginkább hivatkozott módszer, Tauchen módszere. A második módszerről, mely Rouwenhorst nevéhez fűződik, a forrásban belátták, hogy a gyakorlatban ez a pontosabb, ennek ellenére azóta sem népszerű a szakirodalomban.

A továbbiakban végig feltesszük, hogy az idősor oksági, stacionárius, illetve, hogy a független értékű zaj eloszlása normális eloszlású, és ekkor belátható, hogy X_0 is normális eloszlású valószínűségi változó, ezáltal bármely t-re X_t is.

3.2. Tauchen módszer

Tauchen 1986-ban közzétett[2] módszere arra épül, hogy a létrejövő Markov-lánc állapotainak feltételes eloszlásai közelítik az AR(1) eloszlását.

3.2.1. A Markov-lánc előállítása

Adott egy $X_t = \alpha X_{t-1} + \varepsilon_t$ autoregressziós egyenlet. A módszer az α és σ_ε paramétereken felül további kettővel operál:

- $n \in \mathbb{N}^+$: az állapotok száma
- $m \in \mathbb{R}^+$: az állapottér szélességét parametrizálja.

Ezek segítségével készíthető el az $\{X_t\}_{t\in\mathbb{N}}$ Markov-lánc állapottere és sztochasztikus mátrixa.

Az állapottér az $\{x_0, ..., x_{n-1}\}$ halmaz, ahol $x_0 = -m\sigma_X$, $x_{n-1} = m\sigma_X$. A többi állapot egyenletesen osztja fel az $(x_0; x_{n-1})$ intervallumot. A lépésközt h-val jelölve azt kapjuk, hogy $h = \frac{2m\sigma_X}{n-1}$. Ekkor $x_i = -m\sigma_X + i \cdot h$.

Az átmenetvalószínűségek a standard normális eloszlás segítségével adhatóak meg. A standard normális eloszlás eloszlásfüggvénye legyen Φ, a $p_{x_i x_j}$ átmenetvalószínűség pedig

- $\Phi(\frac{x_j - \alpha x_i + h/2}{\sigma_\varepsilon})$, ha $j = 0$
- $1 - \Phi(\frac{x_j - \alpha x_i - h/2}{\sigma_\varepsilon})$, ha $j = n - 1$
- $\Phi(\frac{x_j - \alpha x_i + h/2}{\sigma_\varepsilon}) - \Phi(\frac{x_j - \alpha x_i - h/2}{\sigma_\varepsilon})$, egyébként.

Könnyen belátható, hogy normális eloszlású X_t mellett a kapott átmenetvalószínűségek valóban közelítik az eredeti valószínűségeket. A $p_{x_i x_j}$ valószínűség ugyanis továbbalakítható:
$\Phi(\frac{x_j - \alpha x_i + h/2}{\sigma_\varepsilon}) - \Phi(\frac{x_j - \alpha x_i - h/2}{\sigma_\varepsilon}) = \mathbb{P}(\varepsilon_t < x_j - \alpha x_i + h/2) - \mathbb{P}(\varepsilon_t < x_j - \alpha x_i - h/2) =$
$\mathbb{P}[\varepsilon_t \in (x_j - \alpha x_i - h/2, x_j - \alpha x_i + h/2)]$.
Az AR(1) folyamat állapotterét úgy diszkretizálom, hogy $X_t = x_j$ értsük azt, ha $X_t \in (x_j - h/2, x_j + h/2)$.
Ekkor $p_{x_i x_j} = \mathbb{P}(x_j - h/2 < X_{t+1} < x_j + h/2 \,|\, X_t = x_i) =$
$= \mathbb{P}(x_j - h/2 < \alpha X_t + \varepsilon_t < x_j + h/2 \,|\, X_t = x_i) =$
$= \mathbb{P}(x_j - h/2 < \alpha x_i + \varepsilon_t < x_j + h/2) = \mathbb{P}[\varepsilon_t \in (x_j - \alpha x_i - h/2, x_j - \alpha x_i + h/2)]$.

[2] Az eredeti cikk: Tauchen, G. (1986). Finite state markov-chain approximations to univariate and vector autoregressions. Economics letters, 20.2, 177-181.

3.2.2. A módszer Python-ban

```
def tauchen(n, sigma_e, alpha, m = 3):

    sigma_x = np.sqrt(sigma_e**2 / (1 - alpha**2))

    states = np.linspace(-m * sigma_x, m * sigma_x, n)

    step = (2*m*sigma_x) / (n - 1)
    half_step = step / 2
    P = np.empty((n, n))
    for i in range(n):
        P[i][0] = norm.cdf((states[0]-alpha*states[i]+half_step)/sigma_e)
        P[i][n-1] = 1-norm.cdf((states[n-1]-alpha*states[i]-half_step)/sigma_e)
        for j in range(1,n-1):
            P[i][j] = norm.cdf((states[j]-alpha*states[i]+step/2)/sigma_e)
            P[i][j] -= norm.cdf((states[j]-alpha*states[i]-step/2)/sigma_e)

    mc = qe.MarkovChain(P, states)

    return mc
```

3. ábra. Tauchen módszere Python programozási nyelven

Az első fejezetben használt kódokkal összhangban, Python nyelven programoztam le a módszert, melynek inputja az említett 4 paraméter, viszont m értékét alapértelmezetten 3-ra állítottam be a Quantecon beépített Tauchen függvényéből kiindulva. Ez az [7] forrásban található. A paraméterekből előállítom a *states* állapothalmazt és a P átmenetmátrixot, melyekkel megadom az mc Markov-lánc objektumot, ami a függvény visszatérési értéke.

3.3. Rouwenhorst módszer

Rouwenhorst 1995-ös megközelítése[3] az előzővel ellentétben látszólag meg sem próbálja közelíteni az átmenetvalószínűségeket, ehelyett úgy konstruál egy Markov-láncot, hogy annak várható értéke, szórásnégyzete, feltételes várható értéke, feltételes szórásnégyzete és az elsőrendű autokovarianciája is egyezzen az eredeti folyamat jellemzőivel.

3.3.1. A Markov-lánc előállítása

Adott egy $X_t = \alpha X_{t-1} + \varepsilon_t$ autoregressziós egyenlet. A módszerhez általánosságban további 4 paraméter szükséges, viszont egy részüket inicializálni fogjuk annak érdekében, hogy egyezzenek a momentumok:

- $n \in \mathbb{N}^+$: az állapotok száma
- $\psi \in \mathbb{R}^+$: az állapottér legnagyobb és legkisebb elemének abszolútértéke
- $p, q \in (0, 1)$ kezdetben tetszőleges valószínűségek.

Az állapottér a szimmetrikus, egyenletesen elosztott $\{x_0, ..., x_{n-1}\}$ halmaz lesz, ahol $x_0 = -\psi$, és $x_{n-1} = \psi$. A lépésközt ismét h-val jelölve $h = \frac{2\psi}{n-1}$, és $x_i = -\psi + i \cdot h$.

Az átmenetvalószínűségeket a Θ_n átmenetmátrix tartalmazza, amely az alábbi lépésekkel készíthető el:

- 1. lépés: $n = 2$-re legyen

$$\Theta_2 = \begin{bmatrix} p & 1-p \\ 1-q & q \end{bmatrix}$$

- 2. lépés: $n \geq 3$-ra készítsük el a

$$p \begin{bmatrix} \Theta_{n-1} & \mathbf{0} \\ \mathbf{0}' & 0 \end{bmatrix} + (1-p) \begin{bmatrix} \mathbf{0} & \Theta_{n-1} \\ 0 & \mathbf{0}' \end{bmatrix} + (1-q) \begin{bmatrix} \mathbf{0}' & 0 \\ \Theta_{n-1} & \mathbf{0} \end{bmatrix} + q \begin{bmatrix} 0 & \mathbf{0}' \\ \mathbf{0} & \Theta_{n-1} \end{bmatrix}$$

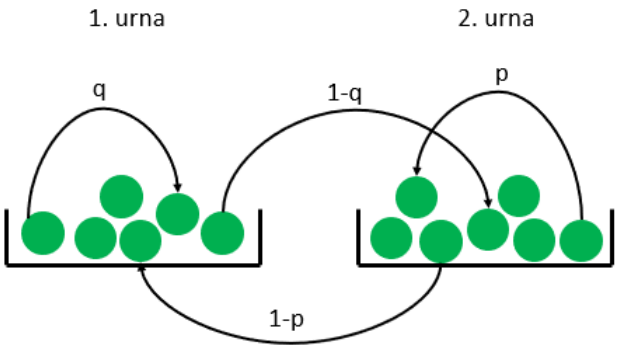
  mátrixot, ahol $\mathbf{0}$ az $(n-1)$ dimenziós nulloszlopvektor.

- 3. lépés: az így kapott mátrix első és utolsó sorát leszámítva minden komponensét osszuk el 2-vel, hogy sztochasztikus mátrixot kapjunk, ez lesz Θ_n.

[3] Az eredeti cikk: Rouwenhorst, K. G. (1995). Asset pricing implications of equilibrium business cycle models. Frontiers of business cycle research, 1, 294-330.

3.3.2. Urnák és golyók

Ezt a módszert emészthetőbbé teszi egy másik megközelítés, melynek egy változata a [3] forrásban is szerepel. Legyen adott 2 urna és $n-1$ golyó, melyek az urnákban vannak elhelyezve. Egy lépés alatt azt értjük, hogy a golyókat egymástól függetlenül vagy helyben hagyjuk, vagy áthelyezzük a másik urnába. Legyen q annak a valószínűsége, hogy egy 1. urnában lévő golyót helyben hagyunk, és legyen p annak a valószínűsége, hogy egy 2. urnában lévőt hagyunk helyben. Jelölje $p_{ij}^{(n)}$ annak a valószínűségét, hogy $n-1$ golyó közül az 1. urnában i golyó volt, és egy lépés után j golyó lett. Ezt a $p_{ij}^{(n)}$ valószínűséget kétféleképpen is ki tudjuk fejezni rekurzívan.



4. ábra. Másik megközelítés: urnák és golyók

Az első esetben rögzítsünk le egy golyót a második urnában. Annak a valószínűsége, hogy ő helyben marad p. Ha a maradék $n-2$ golyót egyben vizsgáljuk, akkor azt mondhatjuk, hogy kétféleképpen lehet a lépés után j golyó az első urnában. A rögzített golyónk vagy helyben marad, és ekkor a maradék golyókra a $p_{ij}^{(n-1)}$ valószínűség kell, vagy átrakjuk az első urnába, és $p_{i,j-1}^{(n-1)}$ valószínűséggel lesz $j-1$ golyó rajta kívül az elsőben. Tehát $p_{ij}^{(n)} = p \cdot p_{ij}^{(n-1)} + (1-p) \cdot p_{i,j-1}^{(n-1)}$.

A második esetben az első urnában rögzítünk le egy golyót. Ekkor ha ez q valószínűséggel helyben marad, akkor a $p_{i-1,j-1}^{(n-1)}$ valószínűség a megfelelő, ha átrakjuk, akkor pedig a $p_{i-1,j}^{(n-1)}$. Tehát $p_{ij}^{(n)} = (1-q) \cdot p_{i-1,j}^{(n-1)} + q \cdot p_{i-1,j-1}^{(n-1)}$.

A Θ_n előállításának lépéseit vizsgálva azt láthatjuk, hogy $(\Theta_n)_{ij}$ valóban egyezni fog $p_{ij}^{(n)}$-nel, mivel az első sor az az eset, amikor az 1. urnában nincsen golyó, ezért az első rögzítésnek nincs itt értelme. Ugyanez igaz az utolsó sorra és a második rögzítésre, a köztes sorokban pedig csak kétszer vesszük ugyanazt a megfelelő valószínűséget, majd elosztjuk 2-vel.

Ha Y_t-vel jelöljük azt, hogy a t. lépés után hány golyó van az 1. urnában, és X_t szokásosan a Markov-lánc t. eleme, akkor mivel az átmenetvalószínűségeik egyeznek, és az állapotereik között lineáris összefüggés van, ezért egy $\{X_t\}_{t\in\mathbb{N}}$ Markov-lánc egy $\{Y_t\}_{t\in\mathbb{N}}$ sorozat lineáris függvényeként áll elő: $X_t = -\psi + \frac{2\psi}{n-1}Y_t$.

3.3.1. Állítás. *A Θ_n átmenetmátrixnak pontosan egy stacionárius eloszlása létezik, ami binomiális eloszlás.*

Bizonyítás: A Θ_n mátrix minden komponense szigorúan pozitív, tehát a 2.4.2 tétel alapján pontosan egy stacionárius eloszlás van. Az urnás megközelítéssel gondolkozva annak keressük az eloszlását, hogy kezdetben az $n-1$ golyóból mennyit milyen valószínűséggel rakjunk az első urnába, ha azt szeretnénk, hogy a lépéseket elvégezve ne változzan ez az eloszlás, ugyanakkora valószínűséggel legyen most i db golyó az elsőben, mint akárhány lépés múlva.

Az $n = 2$ esetre, vagyis amikor 1 golyónk van, egyszerű számolással megkapható a megfelelő eloszlás. Legyen β annak a valószínűsége, hogy a golyónkat az első urnába rakjuk. Ekkor azt szeretnénk, hogy ugyanekkora valószínűséggel legyen 1 golyó az első urnában egy lépés után is: $\beta = \beta q + (1-\beta)(1-p)$, vagyis $\beta = \frac{1-p}{(1-p)+(1-q)}$.

A többi esetre is kiszámolható, hogy a Θ_n-hez tartozó stacionárius eloszlás az $n-1$ rendű, $\frac{1-p}{(1-p)+(1-q)}$ paraméterű binomiális eloszlás. Ennek az az oka, hogy ha a golyókat az elején egymástól függetlenül helyezzük az urnákba, akkor 1 lépés múlva is egymástól független lesz, hogy melyik golyó hol van.

Ez tovább egyszerűsíthető, ugyanis a p, q paramétereket választhatjuk egyenlőnek, a momentumokat ekkor is be fogjuk tudni állítani a megfelelő értékekre. ■

Egyelőre azt mondhatjuk, hogy ez a Markov-lánc még általános, nem használta az autoregressziós egyenlet egyik paraméterét sem. A következő alfejezetben megadjuk, hogy egy ilyen módszerrel előállított Markov-láncnak mik a momentumai, majd beállítjuk úgy a paramétereket, hogy egyezzenek az AR(1) folyamat jellemzőivel.

3.3.3. Momentumok kiszámítása

Az egyszerűség kedvéért tehát legyen $p = q$, ekkor a stacionárius eloszlás paramétere $\frac{1}{2}$. A továbbiakban tegyük fel, hogy a stacionárius eloszlást választjuk kezdeti eloszlásnak, azaz a Markov-lánc stacionárius.

Várható érték: Tehát a stacionárius eloszlás a λ vektor, ahol $\lambda_i = \frac{\binom{n-1}{i}}{2^{n-1}}$.

$$\mathbb{E}[X_t] = \sum_{i=0}^{n-1} \lambda_i x_i = 0,$$

mivel $\lambda_i = \lambda_{n-1-i}$, és $x_i = -x_{n-1-i}$, az állapothalmaz mediánja pedig mindig 0.

Szórásnégyzet:

$$\mathbb{D}^2[X_t] = \sum_{i=0}^{n-1} \lambda_i x_i^2 = \sum_{i=0}^{n-1} \frac{\binom{n-1}{i}}{2^{n-1}} \cdot \left(-\psi + i \cdot \frac{2\psi}{n-1}\right)^2 =$$

$$= \psi^2 + \frac{1}{2^{n-1}} \cdot \frac{4\psi^2}{(n-1)^2} \sum_{i=0}^{n-1} \binom{n-1}{i} i^2 - \frac{1}{2^{n-1}} \cdot \frac{4\psi^2}{n-1} \sum_{i=0}^{n-1} \binom{n-1}{i} i =$$

$$= \psi^2 + \psi^2 \cdot \frac{n}{n-1} - 2\psi^2 = \frac{\psi^2}{n-1}.$$

A célunk az, hogy ez a szórásnégyzet egyezzen az autoregressziós folyamat szórásnégyzetével: σ_x^2. Ez pontosan a $\psi = \sigma_x\sqrt{n-1}$ választással fog teljesülni.

3.3.1. Lemma. *Jelöljük Y_t-vel azt, hogy a t. lépés után hány golyó van az 1. urnában. Ekkor $\forall\, n \geq 2$, $n-1 \geq i \geq 0$-re*

$$\mathbb{E}[Y_{t+1} \,|\, Y_t = i] = (2p-1)i + (n-1)(1-p).$$

$$\mathbb{D}^2[Y_{t+1} \,|\, Y_t = i] = (n-1)p(1-p).$$

Bizonyítás: Szeretnénk kiszámolni $\mathbb{E}[Y_{t+1} \,|\, Y_t = i]$-t, ami azt jelenti, hogy most i db golyó van az első urnában, várhatóan mennyi lesz a következő lépésben. Ekkor az, hogy az első urnában lévő golyók közül mennyi marad ott egy $\mathrm{Bin}(i,p)$ eloszlású valószínűségi változó, mivel minden golyó egymástól függetlenül p valószínűséggel marad. Az pedig, hogy a második urnából mennyi golyó kerül át az elsőbe egy $\mathrm{Bin}(n-1-i, 1-p)$ eloszlású valószínűségi változó. Az $(Y_{t+1} \,|\, Y_t = i)$ valószínűségi változó pedig a konvolúciójuk. Ezek alapján:

$$\mathbb{E}[Y_{t+1} \,|\, Y_t = i] = \mathbb{E}\left[\mathrm{Bin}(i,p)\right] + \mathbb{E}\left[\mathrm{Bin}(n-1-i, 1-p)\right] =$$

$$= ip + (n-1-i)(1-p) = (2p-1)i + (n-1)(1-p).$$

Továbbá:

$$\mathbb{D}^2[Y_{t+1} \,|\, Y_t = i] = \mathbb{D}^2\left[\mathrm{Bin}(i,p)\right] + \mathbb{D}^2\left[\mathrm{Bin}(n-1-i, 1-p)\right] =$$

$$= ip(1-p) + (n-1-i)p(1-p) = (n-1)p(1-p).$$

■

Feltételes várható érték: A 3.3.1 lemmát használva az alábbit kapjuk:

$$\mathbb{E}[X_{t+1} \,|\, X_t = x_i] = \mathbb{E}\left[-\psi + \frac{2\psi}{n-1} \cdot Y_{t+1} \,|\, Y_t = i\right] =$$

$$= -\psi + \frac{2\psi}{n-1} \cdot \mathbb{E}[Y_{t+1} \,|\, Y_t = i] = -\psi + \frac{2\psi}{n-1} \cdot [(2p-1)i + (n-1)(1-p)] =$$

$$= -\psi + 2\psi(1-p) + (2p-1)i \cdot \frac{2\psi}{n-1} = (2p-1) \cdot \left(-\psi + i \cdot \frac{2\psi}{n-1}\right) =$$

$$= (2p-1)x_i,$$

az autoregresszios folyamat feltételes várható értéke αx_i, tehát a $p = \frac{\alpha+1}{2}$ választással érhetjük el, hogy ez a momentum is egyezzen. Látható, hogy p valóban 0 és 1 közötti szám.

Feltételes szórásnégyzet:

$$\mathbb{D}^2[X_{t+1} \,|\, X_t = x_i] = \mathbb{D}^2\left[-\psi + \frac{2\psi}{n-1} \cdot Y_{t+1} \,|\, Y_t = i\right] =$$

$$= \left(\frac{2\psi}{n-1}\right)^2 \cdot \mathbb{D}^2(Y_{t+1} \,|\, Y_t = i) = \frac{4\psi^2}{(n-1)^2}(n-1)p(1-p) =$$

$$= \frac{4\psi^2}{(n-1)}p(1-p).$$

A korábban megválasztott $\psi = \sigma_x\sqrt{n-1}$ és $p = \frac{\alpha+1}{2}$ behelyettesítésekkel pedig

$$\mathbb{D}^2(X_{t+1} \,|\, X_t = x_i) = \frac{4\sigma_x^2(n-1)}{n-1} \cdot \frac{\alpha+1}{2} \cdot \frac{-\alpha+1}{2} = \sigma_x^2(1-\alpha^2) = \sigma_\varepsilon^2.$$

Vagyis a feltételes szórásnégyzetek is egyeznek.

Autokovariancia és autokorreláció: Mivel $\mathbb{E}[X_t] = 0$, ezért $\mathrm{Cov}[X_t, X_{t+1}] = \mathbb{E}[X_t X_{t+1}]$.

$$\mathbb{E}[X_t X_{t+1}] = \sum_{i=0}^{n-1} \lambda_i \mathbb{E}[X_t X_{t+1} \,|\, X_t = x_i] =$$

$$= \sum_{i=0}^{n-1} \lambda_i x_i \mathbb{E}[X_{t+1} \,|\, X_t = x_i] = \sum_{i=0}^{n-1} \lambda_i x_i (2p-1)x_i =$$

$$= (2p-1)\mathbb{D}^2[X_t] = (2p-1)\frac{\psi^2}{n-1} = \sigma_x^2 \cdot \alpha.$$

Tehát az elsőrendű autokovarianca, és ezáltal az autokorreláció is egyezik.

3.3.4. A módszer Python-ban

```
def rouwenhorst(n, sigma_e, alpha):


    sigma_x = np.sqrt(sigma_e**2 / (1 - alpha**2))
    p = (1 + alpha) / 2
    q = p
    psi = sigma_x * np.sqrt(n - 1)
    states = np.linspace(-psi, psi, n)

    def theta_matrix(n, p, q):

        if n == 2:
            theta = np.array([[p, 1 - p], [1 - q, q]])

        elif n > 2:
            p1 = np.zeros((n, n))
            p2 = np.zeros((n, n))
            p3 = np.zeros((n, n))
            p4 = np.zeros((n, n))

            new_mat = theta_matrix(n - 1, p, q)

            p1[:n - 1, :n - 1] = p * new_mat
            p2[:n - 1, 1:] = (1 - p) * new_mat
            p3[1:, :-1] = (1 - q) * new_mat
            p4[1:, 1:] = q * new_mat

            theta = p1 + p2 + p3 + p4
            theta[1:n - 1, :] = theta[1:n - 1, :] / 2

        else:
            raise ValueError("The number of states must be greater than or equal to 2.")

        return theta

    theta = theta_matrix(n, p, q)

    return qe.MarkovChain(theta, states)
```

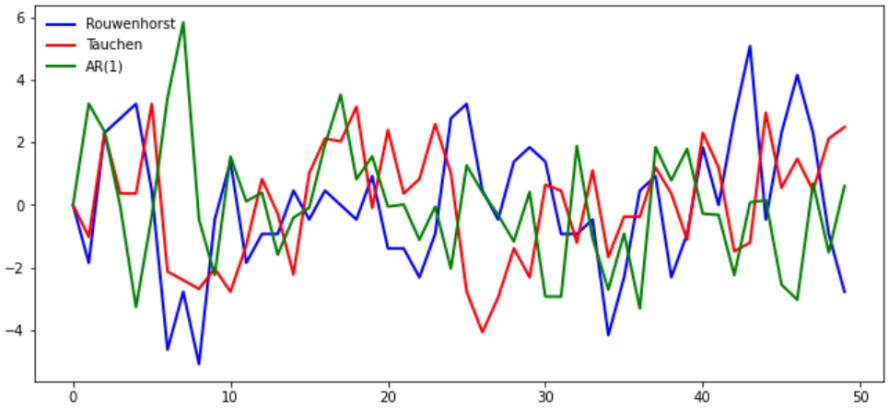
5. ábra. Rouwenhorst módszer Python programozási nyelven

Az előzővel analóg módon az inputok azok a paraméterek, amiket nem a többi paraméterrel határoztunk meg, létrehozzuk a *states* állapothalmazt, a *theta_matrix* függvény segítségével a *theta* átmenetmátrixot, az output pedig a belőlük generált Markov-lánc objektum.

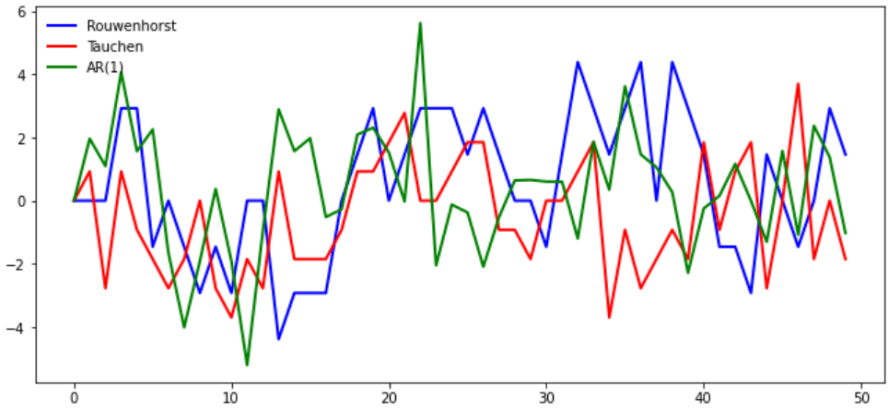
3.4. Vizuális összehasonlítás

Az alábbi két ábrán egy-egy szimulálása látható egy AR(1) folyamatnak, és a hozzá tartozó Tauchen, illetve Rouwenhorst módszerekkel előállított Markov-láncoknak. Az szimulációk közötti különbség az, hogy a második esetben az állapotok számát lecsökkentettem 101-ről 11-re.

```
n = 101
alpha = 0.5
sigma_e = 2
steps = 50

AR = [0]
AR += [alpha*AR[-1] + np.random.normal(0,sigma_e) for i in range(steps-1)]
T = tauchen(n, sigma_e, alpha).simulate(steps,init=0)
R = rouwenhorst(n, sigma_e, alpha).simulate(steps,init=0)

series = [R, T, AR]
labels = ['Rouwenhorst','Tauchen', 'AR(1)']

fig, ax = plt.subplots(1, 1)
for s, label, c in zip(series, labels, ['b','r','g']):
    ax.plot(s, c, lw=2, label=label)
ax.legend(loc='upper left', frameon=False)
plt.show()
```



6. ábra. Idősorok összehasonlítása - 101 állapot



7. ábra. Idősorok összehasonlítása - 11 állapot

A kód elején a paraméterek aktuális értékeit adtam meg. Az autóregresszív folyamatnak külön függvényt nem hoztam létre, egy for ciklus segítségével szimulálom, majd ábrázolom a kapott idősorokat.

Mindkét esetben látszik, hogy a várható érték körül, azaz a 0 körül mozognak a folyamatok, valamint a szórásuk is közel azonos. Az ábrák alapján érdemi különbség nincsen, viszont megemlíthető, hogy több a „vízszintes mozgás„ a második ábrán, mivel ekkor nagyobb a helyben maradás valószínűsége.

4. Eszközárazási feladatok

Az ecopedia.hu[4] honlapon található definíció szerint a pénzügyi eszközök reáleszközökre vonatkozó követelések. Pénzügyi eszköznek minősül minden olyan befektetési eszköz, egyéb tőzsdei termék, valamint minden más eszköz, amelynek forgalmazását szabályozott piacon (például tőzsde) engedélyezték. Emellett pénzügyi eszköznek minősül az a befektetési eszköz is, amely nincsen jelen szabályozott piacon, ám értéke egy szabályozott piacon lévő eszköztől függ.

Az egyszerűség kedvéért a továbbiakban tekintsünk rájuk úgy, mint követelések jövőbeli kifizetésekre. Ezeket a kifizetéseket hívjuk osztalékoknak.

Az utolsó fejezetben, melyet a [6] forrás alapján készítettem, eszközök jelenértékét számolom ki egy olyan modell segítségével, mely Markov-láncokat használ.

4.1. Egyensúlyi ár

Formálisan vegyük a $\{d_t\}_{t\geq 0}$ sorozatot, ami tehát osztalékok egy időbeli sorozata. Egy t-beli ex-osztalékú eszköz a $d_{t+1}, d_{t+2}, ...$ kifizetések követelése, míg egy t-beli halmozott osztalékú eszköz a $d_t, d_{t+1}, ...$ kifizetések követelése. A továbbiakban eszköz alatt mindig ex-osztalékú eszközt értünk, egységnyi idő alatt pedig évet.

A célunk az, hogy a $\{d_t\}_{t\geq 0}$-ről rendelkezésre álló információk alapján meghatározzuk az egyensúlyi árakat, vagyis azokat az értékeket, amelyek esetén a költség és a várható hozam egyenlők. Formálisan azt szeretnénk, hogy a $\{p_t\}_{t\geq 0}$ sorozat elemei minden t-re elégítsék ki a

$$p_t = \beta \mathbb{E}_t[d_{t+1} + p_{t+1}] \tag{1}$$

egyenletet. β a diszkontfaktor, vagyis az egységnyi idő, azaz egy év múlva esedékes pénzáram jelenlegi időpontra számított értékének meghatározásához használt szorzó, $\mathbb{E}_t[y]$ pedig y várható értéke a t. évben rendelkezésünkre álló információk alapján.

Az egyenletekkel tehát azt követeljük meg, hogy ugyanannyira érje meg eladni most az eszközt, mint akárhány év múlva az esedékes osztalékok figyelembevételével.

Általában a pénz elértéktelenédésének mértéke évről évre változik, ezért érdemes megemlíteni azt az általánosabb modellt is, amikor nem a β szorzót használjuk, hanem az $\{m_t\}_{t\geq 0}$ sorozatot, ahol tehát m_t a t. évre vontakozó diszkontfaktor. Ekkor azt követeljük meg az $\{m_t\}, \{d_t\}$ és $\{p_t\}$ sorozatokra, hogy minden t-re

$$p_t = \mathbb{E}_t[m_{t+1}(d_{t+1} + p_{t+1})]. \tag{2}$$

[4] http://ecopedia.hu/penzugyi-eszkoz

A könnyen kezelhetőség érdekében a következő modellekben viszont feltételezzük, hogy a diszkontfaktor konstans, tehát az (1) egyenletből kiindulva fogjuk kiszámolni az egyensúlyi árakat.

Célunk egy Markov-láncokat használó modell megadása, ám először két egyszerűbb modellen végezzük el a számolásainkat, hogy meglegyenek a szükséges intuíciók.

4.2. Konstans osztalék

Legyen minden t-re $d_t = d > 0$, vagyis az osztalék egy előre meghatározott konstans. Gyakorlati példa erre, ha valakinek van egy lakása, mint pénzügyi eszköz, amit bérbe ad, az osztalék pedig a lakbér, ami egy fix összeg bizonyos időközönként fizetve.

Ekkor az (1) egyenletből a várható értéket elhagyva azt kapjuk, hogy

$$p_t = \beta(d + p_{t+1}).$$

Ezt az egyenletet továbbiterálva

$$p_t = \beta(d + p_{t+1}) = \beta d + \beta p_{t+1} =$$

$$= \beta d + \beta^2 d + \beta^2 p_{t+2} =$$

$$\vdots$$

$$= \beta d + \beta^2 d + \beta^3 d + \cdots + \beta^k d + \beta^k p_{t+k}.$$

Mivel az egyenlet jobb oldala lényegében egy mértani sor, ezért amennyiben $\lim_{k\to\infty} \beta^k p_{t+k} = 0$, akkor p_t a $\frac{\beta d}{1-\beta}$-hez konvergál, tehát ez lesz egyensúlyi ár.

$$p_t := \frac{\beta d}{1-\beta} \ \forall\, t \geq 0.$$

A $p_t = \beta(d + p_{t+1})$ egyenletbe visszahelyettesítve könnyedén ellenőrizhető a helyessége.

Mindhárom modellhez létrehoztam egy osztályt Python nyelven. Az első esetben egy eszköz létrehozásához csak a d osztalékot kell megadni, illetve a diszkontfaktort is be lehet állítani, alapértelmezett értéke 0.9. Két függvényt hoztam létre minden esetben, az egyik az egyensúlyi árat számolja ki, a másik pedig ábrázolja a következő adott napszámra az eszköz mutatóit, ez az első esetben nem túl izgalmas.

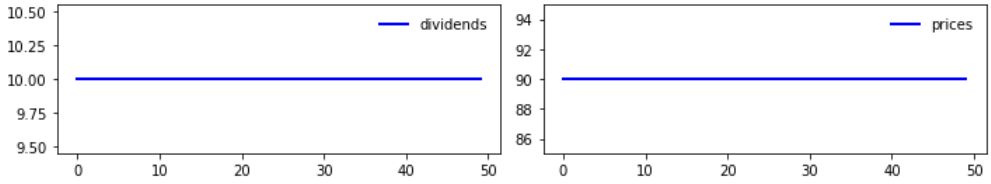
Class for assets of the first type: constant dividends.

```
class modell1:

    def __init__(self, d, β=0.9):

        warning_message = "Inappropriate discount factor."
        assert β < 1 and β > 0,  warning_message

        self.d = d
        self.β = β

    def __str__ (self):
        return 'Asset with dividend ' + str(self.d)

    def eq_price(self):

        p = self.β*self.d/(1 - self.β)
        self.p = p

        return self.p
```

```
    def simulate(self, days):
        modell1.eq_price(self)
        print('The current equilibrium price is ' + str(self.p))

        d_series = [self.d for i in range(days)]

        p_series = [self.p for i in range(days)]


        series = [d_series, p_series]
        labels = ['dividends', 'prices']

        fig, axes = plt.subplots(1, 2)
        fig.set_size_inches(10, 2)
        for ax, s, label in zip(axes.flatten(), series, labels):
            ax.plot(s, 'b-', lw=2, label=label)
            ax.legend(loc='upper right', frameon=False)
        plt.tight_layout()
        plt.show()
```

```
m1 = modell1(10)
m1.simulate(50)
```

```
The current equilibrium price is 90.00000000000001
```



8. ábra. Az 1. modell

4.3. Mértani sorozat, mint osztalék

Legyen minden t-re $d_{t+1} = gd_t$, ahol $0 < g\beta < 1$, vagyis az osztalék mindig ugyanannyi százalékkal nő. $g\beta \geq 1$ esetén az egyensúlyi ár végtelen, hiszen ekkor minden évben többet nő az értéke, mint amennyit romlik az árfolyam. Hétköznapi példa egy fix kamatozású bankbetét, ahol d_t a beáramló kamatok sorozata.

Ahhoz, hogy ebben az esetben ki tudjuk számolni az egyensúlyi árat, kicsit módosítanunk kell az eredeti egyenleten. Legyen

$$v_t := \frac{p_t}{d_t}$$

az ár-osztalék hányados. Ha az (1) egyenlet mindkét oldalát elosztjuk p_t-vel, akkor az alábbit kapjuk:

$$v_t = \mathbb{E}_t[\beta \tfrac{d_{t+1}}{d_t}(1 + v_{t+1})]. \tag{3}$$

Változó osztalék mellett elkerülhetetlen, hogy az ár ne legyen állandó, viszont az eredeti egyenlet jellege miatt gondolhatunk arra, hogy mindig ugyanakkora mértékben változnak, vagyis az ár-osztalék hányados konstans. Ebben a modellben éljünk ezzel a feltételezéssel, tehát $v_t = v$ konstans, így az előző egyenlet arra módosul, hogy $v = \beta \frac{d_{t+1}}{d_t}(1 + v)$, azaz

$$v = \beta g(1 + v).$$

Mivel $\beta g < 1$, ezért v-re a $\frac{\beta g}{1-\beta g}$ pozitív megoldást kapjuk, vagyis az egyensúlyi ár

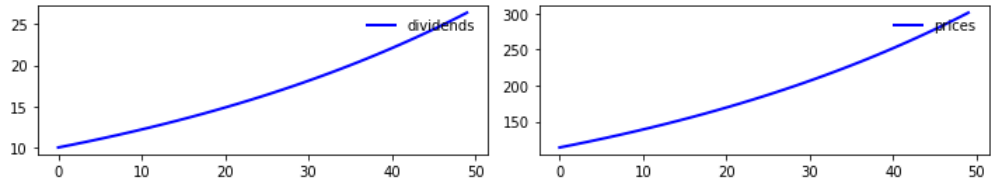
$$p_t := \frac{\beta g}{1 - \beta g} d_t \ \forall\, t \geq 0.$$

A $p_t = \beta(d_{t+1} + p_{t+1})$ egyenletbe visszahelyettesítve könnyedén ellenőrizhető a helyessége.

```python
class modell2:

    def __init__(self, d, g = 1, β=0.9):

        warning_message = "Inappropriate discount factor or dividend."
        assert β*g < 1 and β*g > 0 and β > 0,  warning_message

        self.d = d
        self.g = g
        self.β = β

    def __str__ (self):
        return 'Asset with first dividend' + str(self.d) + ' growing by ' + str(self.g)

    def eq_price(self, t = 1):

        p_t = (self.β*self.g/(1 - self.β*self.g))*self.d*self.g**t

        return p_t


    def simulate(self, steps):
        p = [modell2.eq_price(self, i) for i in range(1,steps+1)]
        self.p = p
        print('The current equilibrium price is ' + str(self.p[0]))

        d_series = [self.d*self.g**i for i in range(steps)]

        p_series = self.p

        series = [d_series, p_series]
        labels = ['dividends', 'prices']

        fig, axes = plt.subplots(1, 2)
        fig.set_size_inches(10, 2)
        for ax, s, label in zip(axes.flatten(), series, labels):
            ax.plot(s, 'b-', lw=2, label=label)
            ax.legend(loc='upper right', frameon=False)
        plt.tight_layout()
        plt.show()
```

```python
m2 = modell2(10, 1.02)
m2.simulate(50)
```

```
The current equilibrium price is 114.1902439024391
```



9. ábra. A 2. modell

Az első esethez teljesen hasonló a programkód, a különbség, hogy itt megadható az elején a g növekedési tényező, alapértelmezett értéke 1, ekkor teljesen megegyezik az első osztállyal. A d paraméter pedig a d_0 osztalék.

4.4. Markov-lánc, mint osztalék

Végső modellünk az előző kettővel szemben véletlent is használ. Legyen az osztalék olyan, hogy

$$d_{t+1} = g_{t+1} d_t.$$

A $\{g_t\}_{t\geq 0}$ sorozat legyen adott

$$g_t = g(X_t),\ t = 1, 2, ...$$

által, ahol $\{X_t\}_{t\in\mathbb{N}}$ egy véges $S = (x_1, ...x_n)$ állapotterű Markov-lánc a P átmenetmátrixszal:

$$p_{xy} = \mathbb{P}(X_{t+1} = y \,|\, X_t = x),\ (x, y \in S).$$

A g pedig egy $S \to \mathbb{R}^+$ függvény.

Lényegében S a világ lehetséges állapotainak halmaza, p_{xy} annak a valószínűsége, hogy az x gazdasági állapot után y következik, a g függvény pedig minden állapothoz hozzárendel egy szorzót, ami növeli, vagy csökkenti az osztalékot. Nyilvánvalóan egy gazdasági válság esetén a g függvény egy 1-nél kisebb számot ad, míg felfelé ívelő gazdasági környezetben akár jóval 1 felettit. A pénzügyi eszközök árának döntő többsége ennek a modellnek megfelelően ingadozik, közrejátszik a véletlen is.

Az előző modellből tudjuk, hogy amennyiben az állapot állandó, az ár-osztalék hányados konstans. Ebből arra következtethetünk, hogy a v_t sorozat elemei csak az aktuális állapottól függnek, vagyis

$$v_t = v(X_t).$$

Ekkor a (3) egyenlet a következőre módosul:

$$v(X_t) = \beta \mathbb{E}_t[g(X_{t+1})(1 + v(X_{t+1}))].$$

X_t-t x-re rögzítve, és a várható érték definícióját használva

$$v(x) = \beta \sum_{y\in S} g(y)(1 + v(y))P(x, y).$$

Továbbalakítva

$$v(x) = \beta \sum_{y\in S} K(x, y)(1 + v(y)),$$

ahol $K(x, y) := g(y)P(x, y)$.

Feltéve, hogy az S halmaz n különböző állapotot tartalmaz, azt mondhatjuk, hogy mindegyikhez tartozik egy ilyen egyenlet, melyeket mátrixalakba írva

$$\mathbb{v} = \beta K(\mathbb{1} + \mathbb{v}).$$

Ahol

- $\mathbb{v}$ a $(v(x_1), ..., v(x_n))'$ oszlopvektor,
- K a $(K(x_i, x_j))_{1 \leq i,j \leq n}$ mátrix,
- $\mathbb{1}$ az n db 1-et tartalmazó oszlopvektor.

Az egyenletet átalakítva:

$$(I - \beta K)\mathbb{v} = \beta K \mathbb{1}.$$

Azt szeretnénk, hogy pontosan egy megoldása legyen az egyenletnek. Ez akkor teljesül, ha $I - \beta K$ invertálható, ami ekvivalens azzal, hogy determinánsa nem 0. [5] Ellenkező esetben az egyenletnek 0 vagy végtelen sok megoldása van, ezzel nem foglalkozunk.

Ekkor a megoldás a

$$\mathbb{v} = (I - \beta K)^{-1} \beta K \mathbb{1}$$

vektor. Vagyis $\mathbb{v}_i = [(I - \beta K)^{-1} \beta K \mathbb{1}]_i = v(x_i)$ az x_i állapothoz tartozó ár-osztalék hányados.

Az eszköz p_t árának kiszámításához tehát ismernünk kell az aktuálisan fennálló állapotot, X_t-t, illetve a legutolsó osztalék értékét, vagyis d_t-t. Ekkor $p_t = d_t v(X_t)$.

Végezetül itt is vizsgáljuk meg egy példán keresztül, hogy hogyan működik a program, és milyen tanulságok vonhatóak le. Fontos kiemelni, hogy minden időpontban az aktuális adatokat használjuk, tehát a $t = 0$ kezdőpillanatban nem tudjuk megmondani az egyensúlyi árat a $t > 1$ értékekre az előző modellekkel ellentétben.

A szimulációhoz szükségünk lesz egy megfelelő Markov-láncra és egy g függvényre. Az előző fejezetben ismertetett Tauchen és Rouwenhorst módszereket használjuk a Markov-lánc előállításához. Tehát egy adott AR(1) folyamatból indulunk ki, amiről feltételezzük, hogy jól modellezi a világ állapotának változását, és elkészítjük hozzá a tanult Markov-láncokat, melyek az AR(1) folyamatot közelítik.

Ugyancsak teljesen hasonló a programkód felépítése, mint az előző esetekben, csak jóval több a paraméter, például a Markov-lánc előállításához használt metódus, és természetes módon hosszabb, összetettebb az egyensúlyi ár számolása.

A vizualizációs függvény az osztalékon és az egyensúlyi áron kívül ábrázolja a gazdasági állapotokat is, illetve a g függvény értékeit, vagyis az osztalék változásához használt szorzókat.

[5] A programkódban a [6] forrás által javasolt módon ellenőrzöm az invertálhatóságot, mivel a determinánsszámolásnál véthet numerikus hibákat.

Class for assets of the first type: Markov growth.

```
class modell3:

    def __init__(self, d, method = tauchen, x0 = 0, g = np.exp, β=0.9):

        warning_message = "Inappropriate discount factor."
        assert β < 1 and β > 0,  warning_message

        self.d = d
        self.x0 = x0
        self.method = method
        self.g = g
        self.β = β

        n = 25
        self.mc = self.method(n, 0.01, 0.96)

        K = self.g(self.mc.state_values) * self.mc.P
        I = np.identity(n)

        warning_message = "Matrix is not invertible."
        assert np.max(np.abs(eigvals(K))) < 1 / β,  warning_message

        v = solve(I - self.β * K, self.β * K @ np.ones(n))
        V = {self.mc.state_values[i]: v[i] for i in range(n)}

        self.V = V

    def __str__ (self):
        return 'Asset with first dividend' + str(self.d) + ' growing by Markov growth '
```

```
    def eq_price(self, x = None, d = None):

        if x == None:
            x = self.x0
        if d == None:
            d = self.d

        p = d*self.V[x]

        return p


    def simulate(self, days, X = True, g = True, d = True, p = True):
        X_series = self.mc.simulate(days, init = self.x0)
        g_series = self.g(X_series)
        d_series = self.d*np.cumprod(g_series)
        p_series = []
        for i in range(days):
            x = X_series[i]
            d = d_series[i]
            p_series.append(modell3.eq_price(self, x, d))

        print('The current equilibrium price is '+ str(p_series[0]))

        verbose = [X, g, d, p]
        series = [X_series, g_series, d_series, p_series]
        labels = ['states','g function','dividends', 'prices']

        fig, axes = plt.subplots(2, 2)
        fig.set_size_inches(10, 4)
        for ax, s, label in zip(axes.flatten(), series, labels):
            ax.plot(s, 'b-', lw=2, label=label)
            ax.legend(loc='upper right', frameon=False)
        plt.tight_layout()
        plt.show()
```

10. ábra. A 3. modell

A paramétereket illetően a [6] forrásban lévőkből indultam ki, ugyanakkor annak érdekében, hogy az invertálhatóság teljesüljön Rouwenhorst módszerénél is, a szórást csökkentenem kellett.

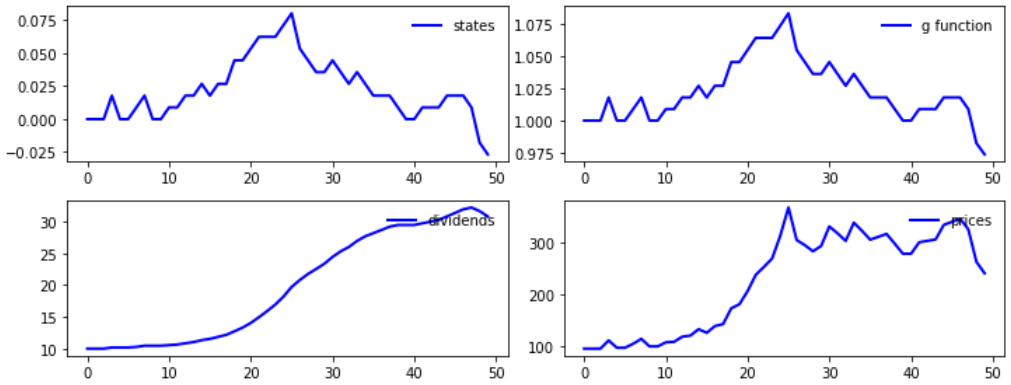
Így végül az autoregresszív folyamathoz, melyet közelítünk az

$$X_t = 0.96X_{t-1} + \varepsilon_t$$

egyenlet tartozik, ahol ε_t szórása 0.01. Az állapotok száma mindkét esetben 25, Tauchen módszerénél az m paraméter pedig az [7] forrás alapján a beépített függvényhez tartozó alapértelmezett érték: 3. A kezdőállapotot mindkét esetben az állapothalmaz mediánjának, 0-nak választottam, a g függvényt pedig az exponenciális függvénynek.

```
m3 = model13(10)
m3.simulate(50)
```

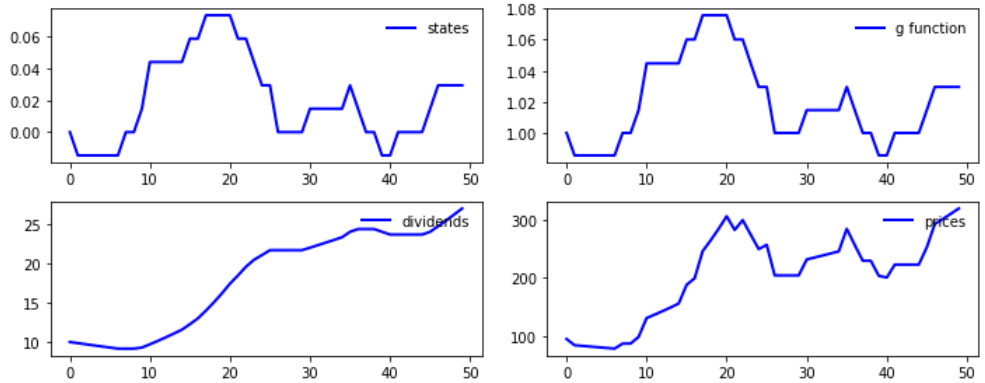
```
The current equilibrium price is 94.5185134172666
```



11. ábra. Egy szimuláció ábrázolása - Tauchen módszer

```
m4 = model13(10, method = rouwenhorst)
m4.simulate(50)
```

```
The current equilibrium price is 94.25223144625915
```



12. ábra. Egy szimuláció ábrázolása - Rouwenhorst módszer

Nem meglepő módon az állapotok változását leíró ábrák nagyon hasonlóak. Ezt tapasztaluk az előző fejezetben is, ugyanis a két módszerrel kapott Markov-láncok egy-egy szimulációját láthatjuk. Egy különbség emelhető ki az ábrák alapján, Rouenhorst módszerénél a helybenmaradás valószínűsége magas lehet, ebből kifolyólag gyakori a vízszintes mozgás.

Ugyanakkor ami miatt mégis előfordulhatnak jelentős különbségek a szórásban, a csökkenések számában az az átmenetmátrixaik közti különbségből adódik. Ezáltal ugyanis az ár-osztalék hányados értékei jelentősen is eltérhetnek. Ezek felszínes vizsgálatára a Python Pandas könyvtárának segítségével készültek el az alábbi táblázatok.

Model3 - Tauchen method

	0	1	2	3	4	5	6	7	8	9	10	11
state values	-0.11	-0.10	-0.09	-0.08	-0.07	-0.06	-0.05	-0.04	-0.04	-0.03	-0.02	-0.01
g function	0.90	0.91	0.91	0.92	0.93	0.94	0.95	0.96	0.96	0.97	0.98	0.99
price-dividend ratio	4.96	5.13	5.35	5.61	5.90	6.22	6.56	6.94	7.36	7.81	8.30	8.85
probability	1.7e-03	2.7e-03	5.2e-03	9.0e-03	1.5e-02	2.3e-02	3.4e-02	4.7e-02	6.1e-02	7.4e-02	8.6e-02	9.4e-02

	12	13	14	15	16	17	18	19	20	21	22	23	24
state values	0.00	0.01	0.02	0.03	0.04	0.04	0.05	0.06	0.07	0.08	0.09	0.10	0.11
g function	1.00	1.01	1.02	1.03	1.04	1.05	1.06	1.06	1.07	1.08	1.09	1.10	1.11
price-dividend ratio	9.45	10.12	10.85	11.66	12.57	13.57	14.67	15.90	17.23	18.67	20.17	21.62	22.80
probability	9.7e-02	9.4e-02	8.6e-02	7.4e-02	6.1e-02	4.7e-02	3.4e-02	2.3e-02	1.5e-02	9.0e-03	5.2e-03	2.7e-03	1.7e-03

13. ábra. Állapotok táblázat - Tauchen módszer

Model3 - Rouwenhorst method

	0	1	2	3	4	5	6	7	8	9	10	11
state values	-0.17	-0.16	-0.15	-0.13	-0.12	-0.10	-0.09	-0.07	-0.06	-0.04	-0.03	-0.01
g function	0.84	0.85	0.86	0.88	0.89	0.90	0.92	0.93	0.94	0.96	0.97	0.99
price-dividend ratio	3.54	3.77	4.02	4.31	4.62	4.98	5.38	5.84	6.37	6.97	7.67	8.48
probability	6.0e-08	1.4e-06	1.6e-05	1.2e-04	6.3e-04	2.5e-03	8.0e-03	2.1e-02	4.4e-02	7.8e-02	1.2e-01	1.5e-01

	12	13	14	15	16	17	18	19	20	21	22	23	24
state values	0.00	0.01	0.03	0.04	0.06	0.07	0.09	0.10	0.12	0.13	0.15	0.16	0.17
g function	1.00	1.01	1.03	1.04	1.06	1.08	1.09	1.11	1.12	1.14	1.16	1.17	1.19
price-dividend ratio	9.43	10.54	11.86	13.43	15.32	17.59	20.35	23.71	27.83	32.92	39.23	47.09	56.95
probability	1.6e-01	1.5e-01	1.2e-01	7.8e-02	4.4e-02	2.1e-02	8.0e-03	2.5e-03	6.3e-04	1.2e-04	1.6e-05	1.4e-06	6.0e-08

14. ábra. Állapotok táblázat - Rouwenhorst módszer

A két táblázat módszerenként tartalmazza az állapotokat, a g függvény lehetséges értékeit, az állapotokhoz tartozó ár-osztalék hányadosokat, illetve a Markov-láncok stacionárius eloszlásai alapján az állapotok valószínűségeit.

Az állapothalmazok, és ezáltal a g függvény értékei és a hányadosok között is jelentős különbség figyelhető meg, Rouwenhorst módszere szélsőségesebb. Ezt azért nem próbáltam meg átjavítani a paraméterek változtatgatásával, mivel a szélsőséges állapotokhoz jóval alacsonyabb valószínűségek tartoznak, mint Tauchen módszerénél. Így a két metódussal nagyjából ugyanazokat az árak kaphatóak, viszont Rouwenhorst módszerének használatakor számítani lehet extrémebb ugrásokra. Ez összhangban van a vizualizáció során látottakkal.

5. Összegzés

Szakdolgozatom első fejezetében megismerkedtünk a Markov-láncokkal. Volt szó átmenetmátrixról, periodicitásról, irreducibilitásról, stacionárius eloszlásról, valamint az ezekhez tartozó, a Python programozási nyelv Quantecon kiegészítő csomagjából elérhető függvényeket is bemutattam.

A második fejezet javarészt az autoregresszív folyamatokról szólt. Megadtam többek között a feltételes várható értékét, kovarianciáját, majd bemutattam két metódust, mely Markov-láncokkal közelíti az AR(1) folyamatot. Névlegesen a Tauchen és Rouwenhorst módszereket, melyekről meg kell említeni, hogy mennyire különböző ötletből indulnak ki ugyanarra a célra. Leprogramozásuk után egy vizualizációs példa segítségével rábólinthattunk arra, hogy valóban összhangban vannak.

Az utolsó fejezetben pedig kiszámoltam először elméletben, majd gyakorlatban egy eszköz egyensúlyi árát, amihez alkalmaztam a második fejezetben megismert módszereket.

Ugyan arra nem jöttem rá, hogy a Markov-lánc világ ideálisabb-e a mi világunknál, de egy kellemes utazást tölthettem el benne a tanév alatt, mely során megismerkedtem új matematikai szemléletekkel, és fejlesztettem programozási készségeimet.

Hivatkozások

[1] Csiszár Villő:
Diszkrét és folytonos idejű Markov-láncok, egyetemi jegyzet

[2] Kopecky, Karen A. and Suen, Richard M. H.:
Finite state Markov-chain approximations to highly persistent processes,
Review of Economic Dynamics, 2010, 13.3: 701-714

[3] Lkhagvasuren, D., Bataa, E.:
Finite-State Markov Chains with Flexible Distributions,
Computational Economics, 2022, 1-34.

[4] Márkus László:
Idősorok I. - 1. rész, 2019, 2-7 és 29-38

[5] Thomas J. Sargent, John Stachurski:
Quantitative economics with Python: Finite Markov Chains

[6] Thomas J. Sargent, John Stachurski:
Quantitative economics with Python: Asset Pricing: Finite State Models

[7] QuantEcon Developer Team:
Source code for quantecon.markov.approximation