XI. Erdélyi Tudományos Diákköri Konferencia

Matematika szekció
Kolozsvár, 2008. május 23-24.

Cash management optimizáció

Témavezetők
Dr. Soós Anna, előadótanár
BBTE, Matematika-Informatika Kar,
Numerikus analízis tanszék
Dr. András Szilárd, adjunktus
BBTE, Matematika-Informatika Kar,
Differenciálegyenletek tanszék

Szerző
Sipos Kinga
BBTE, Matematika-Informatika Kar,
Komputacionális matematika szak, I év

Tartalomjegyzék

1. fejezet

Indíték

A bankkártya, illetve a hitelkártya használata egyre elterjedtebb, ennek ellenére még nagyon sok kereskedelmi tranzakciót bonyolítunk készpénz segítségével. A kisebb pénzösszegeket mozgosító ügyletek legnagyobb részében a készpénz a fizetőeszköz és az összes kereskedelmi tranzakciót tekintve is a legnagyobb népszerűségnek a készpénz örvend. Hollandiában az évi készpénzügyletek száma kb. 7 billióra tehető, ami azt jelenti, hogy minden ember átlagosan naponta $1 - 1\frac{1}{2}$ alkalommal fizet készpénzzel [9]. Következésképpen, a készpénzzel mindannyiunknak mindennapi kontaktusa van.

A mindennapi készpénz ügyletek kivitelezéséhez szükséges címletek és pénzérmék biztosításához bankok és kereskedők egy költséges pénzelosztó rendszerben vesznek részt. Ennek a rendszernek a fenntartása rengeteg pénzbe kerül. A kereskedőknek biztosítaniuk kell többfajta fizetési lehetőséget üzleteikben, ezen kívül számukra költséget jelent a felhalmozódott készpénz előkészítése a bankba való betételre (a pénz megszámolása, a bankjegyek és pénzérmék csomagolása). A bankok kiterjedt ATM-hálózatokat üzemeltetnek és sok fiókot létesítenek, hogy kielégítsék klienseiket. Alapvető feladatuk az igényeknek megfelelő mennyiségű készpénzt biztosítani. A Nemzeti Banknak van egy plusz feladata a többi bankhoz képest: a pénzgyártás. Ugyancsak neki kell gondoskodnia arról, hogy szükség esetén megfelelő mennyiségű készpénzt tudjon szolgáltatni a többi bank számára és ő a felelős a használt címletekről való döntésben. A cash számos költséget generál úgy a bankok, mint a kereskedők számára.

Hollandiában például 2004 körül ezen rendszer fenntartására költött pénzmennyiséget 2 billió euróra becsülték [9]. Ez az összeg elég nagy még egy ország számára is és társadalmi szinten is elég sok mindenkit érint ahhoz, hogy megpróbálkozzunk a csökkentésével. A cash management pontosan erre törekszik.

2. fejezet

Banki költségek optimizálása

A készpénz ügyletekkel járó költségek nagy része a bankokat illeti, ezért a bankok szempontjából közelítjük meg a problémát, azaz a bankok készpénz ügyletekből származó költségeinek csökkentésére koncentrálunk. Lássuk, pontosan melyek ezek a költségek.

2.1. Banki költségek

A bank kiadásainak két fajtáját különböztetjük meg:

1. alkalmi költségek;
2. állandó költségek.

Alkalmi költségek az egyszeri, meg nem ismétlődő kiadások, például egy új bankfiók létesítése. Az állandó költségek a következő fajták lehetnek:

a) **holding cost** [1], az a költség, ami a pénz tárolásához szükséges (a pénztároló helyiség/ATM bizonyos értékre való bebiztosítása, az érték készpénzben való tárolásából adódó kamatveszteség, logisztikai szállítás);

b) **delivery cost**, a nem logisztikai, hanem valami más céllal (például ATM feltöltése) történő szállítások költségei;

c) **interbank trade cost**, a pénzvásárlás költsége;

d) **cash processing cost**, a pénzfeldolgozás költsége (pénzszámlálás, csomagolás, nyilvántartás);

[1] Az angol megnevezést azért tüntettem fel, mert nem mindig van pontosan magyar megfelelő a különböző költségek kifejezésére, illetve szakirodalmat is inkább angolul találunk.

e) **administrative cost**, adminisztratív költségek (a banki személyzet fenntartása).

A készpénz egyetlen bevételi forrást tesz lehetővé a bankok számára, éspedig azt, ami a pénzeladásból származik (**interbank trade income**).

Megjegyzés.

1. Gyakran problémát jelent elkülöníteni a holding costot a többitől, ez nehézségeket okoz a pontos költségek meghatározásában, ami az optimizációt bonyolítja, teszi pontatlanná vagy lehetetlenné.

2. Nehéz felmérni, mennyi időt vesz igénybe egy ember számára bizonyos mennyiségű pénz megszámolása és csomagolása. Ez azt eredményezi, hogy nem állapítható meg pontosan a pénzfeldolgozáshoz szükséges emberek száma. Következésképpen a pénzfeldolgozás költségének csökkentése is bonyodalmakba ütközik.

2.2. Az optimizáció szintjei

A bank kiadásainak optimizációja során 4 egymásra épülő szinttel kell számolnunk:

1. banki automaták (ezután csak ATM-ek), amelyekből csak felvenni lehet pénzt;

2. ATM-ek, amelyek pénzfelvételre, -betevésre (esetleg -váltásra) egyaránt alkalmasak;

3. bankfiókok, amelyek üzemeltetik a saját automatáikat és gondoskodnak a megfelelő menynyiségű készpénzről, illetve eladják a felhalmozódott készpénzt;

4. bankok, ők döntenek új bankfiókok létrehozásáról vagy régiek megszüntetéséről, ugyancsak ők határoznak hosszú távú stratégiák tekintetében, illetve ezen a szinten történnek a nagyszabású készpénz-eladások, -vásárlások és a befektetések.

2.3. Az optimizáció mikéntje

Az optimizációt úgy képzeljük el, hogy a bank történeti adatatainak (2-3 év, akár több is, ha rendelkezésünkre áll) segítségével előrejelzést szerkesztünk a jövőbeli adatokra, ilyen adatok például az egyes cash pointoknál (ATM, bankfiók, bank) felvett, letétbe helyezett vagy felváltott napi pénzösszeg, a pénzvásárlás, illetve eladás dátuma és az összegek nagysága. Az előrejelzések és esetleg az aktuális piaci alakulás alapján egy algoritmust tervezünk, amely képes a közelebbi és távolabbi jövőre vonatkozó döntéseket hozni:

megmondja, mikor mennyi pénzzel kell feltölteni az ATM-eket, mikor érdemes a felgyűlt készpénzt eladni, illetve megtartani, eldönti mikor érdemes egy ATM-t vagy bankfiókot üzemeltetni, mikor indokolt egy új ATM vagy bankfiók létrehozása és mikor kell megszüntetni egy cash pointot.

3. fejezet

Az ATM költségeinek optimizálása

A banki költségek optimizációjának első szintjét probáljuk megoldani, azaz az ATM üzemeltetésének optimizációjával fogunk foglalkozni.

Egy ATM működésének hatékonyságát az ATM kiürítésekor a felhasználatlan készpénz és a feltöltendő pénzmennyiség arányával mérik. Ez a hatékonysági mutató Romániában 15%, ami más nyugati országokban 7% körüli. Ennek a különbségnek a magyarázata, hogy nálunk semmiféle software-t nem használnak az ATM-ek üzemeltetésének optimizálására a kisebb hatékonysági mutatóval rendelkező országokkal ellentétben. A Raiffeisen bank rendelkezett egyedül egy kezdetleges software-rel, de rövid (valószínűleg nem túl hatékony) használat után lemondtak róla és egy külső vállatot kértek fel ATM-eik hatékony működtetésére (idegen kifejezéssel outsourcing-oltak). A többi bank nálunk pár saját alkalmazottját bízza meg ezzel a feladattal, akik mindenféle cél-software nélkül hozzák meg döntéseiket.

3.1. Konkrét feladat megfogalmazása

Egy ATM-et a következő költségek terhelnek:

a) **holding cost**, az ATM bizonyos értékre való bebiztosítása, az érték készpénzben való tárolásából adódó kamatveszteség;

b) **delivery cost**, az ATM feltöltésének költségei.

Megjegyzés. Az ATM nem rendelkezik sokfajta költséggel és ezek egyértelműen meghatározhatóak és elkülöníthetőek a teljes bank költségeivel ellentétben, amelyek gyakran egymásba olvadnak vagy nem határozhatóak meg egyértelműen. Így az ATM optimizációja átláthatóbb a bank optimizációjánál.

Az előző költségek közül az egy ATM bebiztosításának költsége elég hosszú ideig állandó, ezért mi az optimizáció során rögzítettnek tekintjük, habár elképzelhető olyan optimizáció is, amely ezt is figyelembe veszi, és eldönti, milyen értékre érdemes bebiztosítani egy ATM-et annak függvényében például, hogy mennyi pénzmennyiséggel szokták feltölteni (ez nem mindig egyezik meg az ATM teljes kapacitásával).

A maradék költség, ami optimizálásra szorul, az érték készpénzben való tárololásából adódó kamatveszteség (ami minden nap végén az ATM-ben megmaradt pénzösszeg napi kamatja) és az ATM feltöltésének költségei. Az első költség (tulajdonképpen veszteség) annál kisebb, minél kevesebb pénzmennyiség marad nap végén az ATM-ben. Ez alapján célszerű az ATM-et minél kisebb pénzmennyiségekkel feltölteni. A második költség csökkentése az elsőével ellentétes stratégiát követel: a szállítási költség annál kisebb, minél kevesebb alkalommal kell feltölteni az ATM-et (a szállítás díja ugyanis független az ATM-be szállított pénz mennyiségétől, csak a szállítás/feltöltés alkalmainak számától függ). Mindkét költség a feltöltések időpontjától és nagyságától függ. Tehát a pénzszállítások beütemezése a feladat úgy, hogy az előző két ellentétes irányú szállítási stratégiát ,,kibékítsük", miközben soha nem engedjük meg, hogy kiürüljön az ATM (az ATM kiürülése egy forgalmas helyen megengedhetetlen, ez nagyon csökkentené az ATM image faktorát).

A fentiek összegzéseképpen, a feladat a feltöltések időpontjának és nagyságának meghatározása, amely a következő két alapelv követésével történik, miközben a költségek minimalizálására törekszünk:

1. az ATM soha ne maradjon üres;

2. az ürülőfélben levő ATM tartalma minél kisebb legyen feltöltéskor.

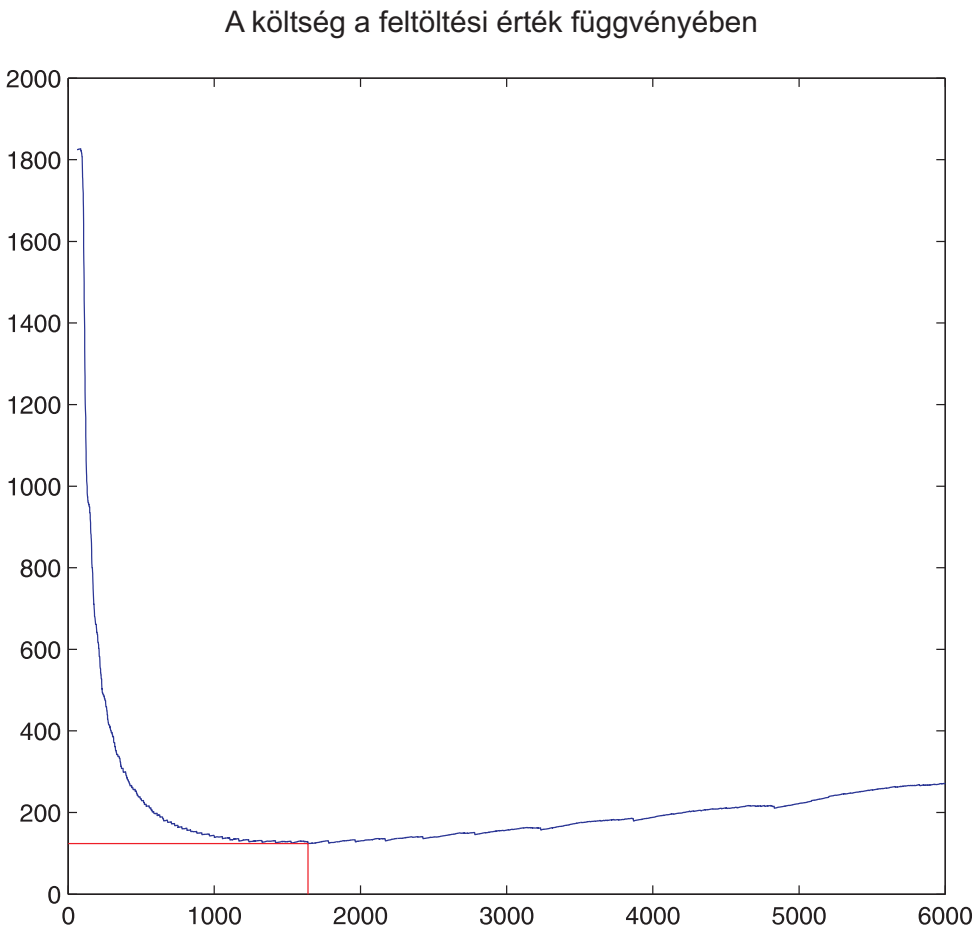
3.2. Az optimizáló algoritmus

Ismert az ATM feltöltésének az ára, az éves kamatláb és az ATM-nek egy maximális kapacitása. Kíváncsiak vagyunk, hogyan programozzuk be a feltöltéseket, hogy az ATM-et minél hatékonyabban működtessük.

Első lépésben rögzítjük az ATM-hez kiszállított pénzmennyiséget, azaz feltételezzük, hogy mindig ugyanannyi készpénzzel töltjük fel az ATM-et, majd megszerkesztünk egy költségfüggvényt, amely ha ismert a napi pénzfelvétel és adott a feltöltések nagysága/értéke, beütemezi a szállításokat a két alapelvnek megfelelően, és kiszámolja az így adódó költségeket, ami egyben a függvény által visszatérített érték.

A szállítások időzítése nyilvánvalóan a jövőre vonatkozik, utólag - pontosan ismerve az ATM-ből naponta felvett összegeket - könnyű megmondani, hol tévedtünk: mikor kellet volna plusz pénzszállítást beütemezni és mikor történt túl hamar az ATM feltöltése. A múlt ilyen szempotból érdektelen, nekünk a jövőbeli szállításokat kell beütemezni[1]. A beütemezés, mint láttuk a költségfüggvény kiértékelésekor történik, ez viszont lehetetlen a napi pénzfelvétel ismerete nélkül, a jövőre vonatkozó napi pénzfelvétel pedig ismeretlen. Ezen nehézség áthidalására a történeti adatokból, idősor-modellek (ARMA, ARIMA, GARCH [16]) illesztésével adunk egy évre előre vonatkozó becslést.

Következő lépésben megvizsgáljuk, hogyan változik a költségfüggvény értéke a feltöltött pénzmennyiség változtatásával. Könnyű belátni, hogy ez a függvény mindig pontosan egy minimummal rendelkezik. Ezt szemlélteti a következő ábra:



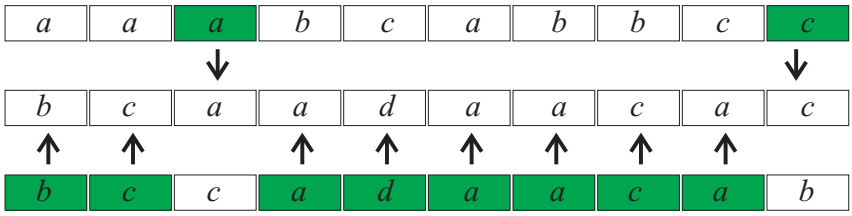
Megkeressük, milyen feltöltési mennyiségre éri el a költségfüggvény a minimumát, és a

[1]Az előre való beütemezés azért fontos, mert ha észrevesszük, hogy kiürült az ATM vagy hamarosan ki fog ürülni, de még nincs kilátásban beütemezett feltöltés, akkor úgynevezett prompt (azonnali) szállítást kell kérni a szállítócégtől, ami sokkal többe kerül, mint a beütemezett. Pontosabban a legalább két nappal előre jelzett ATM-feltöltés fix összegbe kerül, különben minél kevesebb idő van hátra a szállítás pillanatáig, annál többe kerül a szállítás.

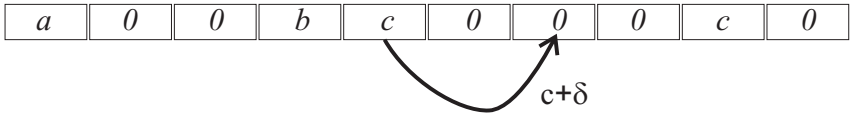
feltöltések nagyságát erre az értékre állítjuk, majd az előrejelzések és a feltöltési érték ismeretében beütemezünk egy évre a szállítást. Azért van szükség ilyen hosszútávú beütemezésre, mert ahhoz, hogy megkeressük a költségfüggvény minimumát a feltöltési érték szerint, elég sok feltöltésre van szükség, ami $1-2$ hét vagy hónap alatt nem következik be.

Tovább szeretnénk csökkenteni a költségeket, ezért a következő genetikus algoritmust építjük fel. Legyen egy genom egy n (365) komponensű vektor, amelynek a j-edik eleme a j-edik napon történő feltöltés mennyisége (tehát egy olyan vektorra kell számítanunk, amely csak pár nullától különböző komponenssel rendelkezik). Minden egyedet jellemez a genom. A populációnknak m (1000) egyede van. Ezek kezdetben azonosak, éspedig mind azzal a genommal rendelkeznek, amely az eddig végrehajtott klasszikus optimizáció eredményeként nyert feltöltési beütemezéseknek felel meg. A genomokon a következő operációkat értelmezzük:

- keresztezés (két különböző A és B genomból megkapjuk a C genomot úgy, hogy C_i valamilyen valószínűségi változó szerint megegyezik az A_i-vel vagy a B_i-vel)



- mutáció (egy genom minden elemét elmozdítjuk egy másik pozicióra a környezetében és értékét is megváltoztatjuk egy bizonyos korlátok közötti véletlen számmal)



Egy lépés több mutációból és keresztezésből áll. Kiválasztjuk az egyedek p_1 százalékát, ők fognak mutációt szenvedni és p_2 százalékuk vesz részt keresztezésben. Ezen operációk során keletkezett új egyedeket is hozzávesszük a populációnkhoz ideiglenesen, de a régiek is megmaradnak. A mutációk és keresztezések után minden egyedet kiértékelünk és csak az m ,,legjobb" (legkisebb költséget generáló) egyedet őrizzük meg a populációban. Ezután a folyamatot megismételjük.

A fenti genetikus algoritmusban, ha mutáció vagy kereszteződés során létrejönnek olyan egyedek, amelyek nem tartalmaznak elegendő feltöltést, azaz ha a nekik megfelelő

pénzszállítási tervet követve (esetleg többször is) kiürülne az ATM (amit mi nem engedhetünk meg), akkor az egyedet úgymond regularizáljuk, vagyis az ATM kiürülését megelőző napra egy megfelelő értékű feltöltést időzítünk (ez a módosítás az egyedek kiértékelésekor történik). Enélkül a módosítás nélkül a genetikus algoritmussal nem jutunk használható eredményhez.

Összegezve, az algoritmus 3 kulcslépése:

1. előrejelzés;

2. a feltöltés összege szerinti minimalizáció;

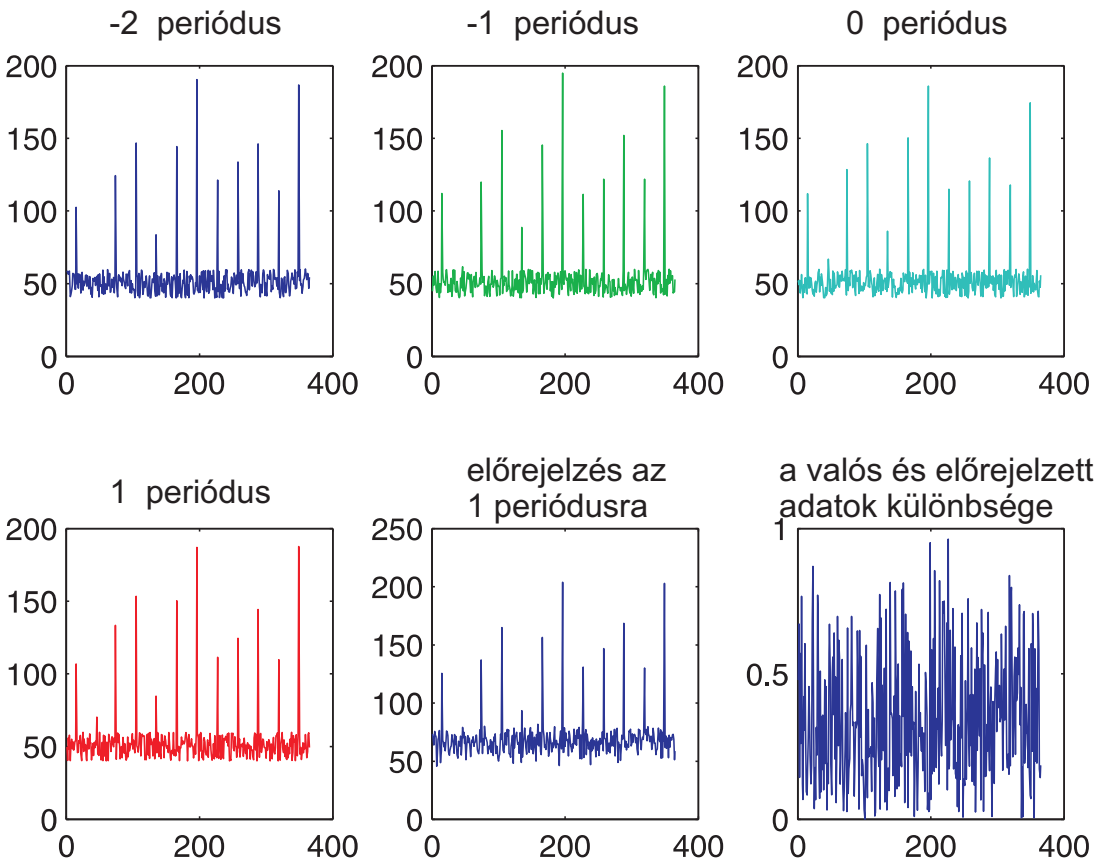
3. optimizálás genetikus algoritmussal.

Az algoritmust minden nap lefuttatjuk (úgy, hogy az előző napi adatokat is a történeti adatokhoz csatoljuk), és az általa megadott előrejelzést - bár éves - csak a következő $1-2$ napra alkalmazzuk. Lehet, hogy az hosszabb távon is még megbízható (5-7 nap), de egy évre biztosan nem. Így a feltöltés mennyisége sem lesz tulajdonképpen fix egész év alatt, az algoritmus minden futtatásakor újraszámolódik.

3.3. Az algoritmus által biztosított hatékonysági mutató

Szimulációinkban 3 évre visszamenő történeti adatokat használtunk ($-2, -1, 0$ periódus) és előrejeleztük az aktuális évi naponta felvett pénzmennyiségeket. Az alábbi ábra első 3 grafikonján láthatóak ezen történeti adatok, a 4-dik grafikon az aktuális évi napi pénzfelvételeket tartalmazza, az 5-dik grafikon az aktuális évre az előrejelzéseket[2] és az utolsó a reális aktuális évi és az előrejelzés közti különbséget.

[2]Itt az előrejelzésre a megfelelő nap három évi átlagát használtuk.



Megjegyzés. A pénzfelvételek nagyságát egy egyenletes eloszlású és több normál eloszlású valószínűségi változó összegeként hoztuk létre. A különböző várható értékű normál eloszlásokra azért volt szükség mert több csúcsot/kiugrást szerettünk volna generálni a felvett pénzösszegek között, amivel mondjuk egy az ATM-hez közeli gyár fizetésnapjait modellezhetjük.

A kalsszikus optimizálásra végzett 1000 szimuláció során a hatékonysági mutató (azaz az ATM-be betett, fel nem használt pénz százalékos aránya) mindig $1,5\%$ és 5% közötti volt és ennek az 1000 évre számolt átlaga pedig 2.73.

Szimulációnkban a genetikus algoritmus használatával sikerült további 10%-ot javítanunk a klasszikus algoritmus segítségével már lecsökkentett költségeken. Egyetlen ATM esetén a klasszikus optimizáló algoritmus 2 másodpercet futott, míg a genetikus algoritmus kevés számú iteráció (40) és kis populáció (100 egyed) esetén is 30 másodpercet igényel.

Megjegyzés. A szimulációk eredményei azt mutatják, hogy az algoritmus használható, de konkrét történeti adatait egyetlen ATM-nek sem ismerjük, így valós tapasztalataink nincsenek a hatákonyságról. Adatainkat, amelyekre lefuttattuk szimulációinkat adott eloszlás szerint generáltuk. Kérdés, hogy ez lényegileg befolyásolja-e az algoritmus eredményeit.

4. fejezet

Az előrejelzés matematikai háttere

1. Értelmezés. *Tekintsük egy* $\{Z(\omega,t) : t \in \mathbb{Z}\}$ *sztochasztikus folyamat valószínűségi változóinak egy véges halmazát:* $\{Z_{t_1}, Z_{t_2}, \ldots, Z_{t_n}\}$. *A* $Z_{t_1}, Z_{t_2}, \ldots, Z_{t_n}$ *valószínűségi változók* ***n*-dimenziós együttes eloszlásfüggvénye** $F_{Z_{t_1},\ldots,Z_{t_n}} : \mathbb{R}^n \to \mathbb{R}$, *amelyre*

$$F_{Z_{t_1},\ldots,Z_{t_n}}(x_1,\ldots,x_n) = P(\omega : Z_{t_1} \leq x_1, \ldots, Z_{t_n} \leq x_n), \forall (x_1,\ldots,x_n) \in \mathbb{R}^n.$$

2. Értelmezés. *Egy sztochasztikus folyamatról azt mondjuk, hogy* **elsőrendű stacionárius folyamat eloszlásban**, *ha az egydimenziós eloszlásfüggvényei időtől függetlenek, azaz* $F_{Z_{t_1}}(x_1) = F_{Z_{t_1+k}}(x_1), \forall t_1, k \in \mathbb{Z}, x_1 \in \mathbb{R}$; **másodrendű stacionárius folyamat eloszlásban**, *ha a kétdimenziós eloszlásfüggvényei időtől függetlenek, azaz* $F_{Z_{t_1},Z_{t_2}}(x_1,x_2) = F_{Z_{t_1+k},Z_{t_2+k}}(x_1,x_2), \forall (t_1,t_2) \in \mathbb{Z}^2, k \in \mathbb{Z}, (x_1,x_2) \in \mathbb{R}^2$; ***n*-ed rendű stacionárius folyamat**, *ha*

$$F_{Z_{t_1},\ldots,Z_{t_n}}(x_1,\ldots,x_n) = F_{Z_{t_1+k},\ldots,Z_{t_n+k}}(x_1,\ldots,x_n), \tag{4.1}$$
$$\forall (t_1,\ldots,t_n) \in \mathbb{Z}^n, k \in \mathbb{Z}, (x_1,\ldots,x_n) \in \mathbb{R}^n.$$

3. Értelmezés. *Egy sztochasztikus folyamat* **szigorúan stacionárius**, *ha (4.1) igaz bármely* $n = 1, 2, \ldots$.

Megjegyzés. Ha (4.1) igaz $n = m$-re, akkor ugyancsak igaz lesz minden $n \leq m$ természtes számra is, mert az m-ed rendű eloszlásfüggvény minden alacsonyabb rendű eloszlásfüggvényt meghatároz.

4. Értelmezés. *Egy adott* $\{Z_t : t \in \mathbb{Z}\}$ *valós értékű folyamat esetén* **a folyamat várható érték függvénye**

$$\mu_t = E(Z_t),$$

a folyamat varianciafüggvénye

$$\sigma_t^2 = Var(Z_t) = E(Z_t - \mu_t)^2,$$

Z_{t_1} *és* Z_{t_2} **kovaranciája**

$$\gamma(t_1, t_2) = E(Z_{t_1} - \mu_1)E(Z_{t_2} - \mu_2),$$

Z_{t_1} *és* Z_{t_2} **korrelációja**

$$\rho(t_1, t_2) = \frac{\gamma(t_1, t_2)}{\sqrt{\sigma_{t_1}^2}\sqrt{\sigma_{t_2}^2}}.$$

Megjegyzés. Egy szigorúan stacionárius folyamat, amelynek első két abszolút momentuma véges, konstans (időtől független) várható érték függvénnyel és varianciafüggvénnyel rendelkezik, valamint a kovariancia- és korrelációs függvénye csak az időbeli $(t_2 - t_1)$ távolságtól függ.

A fenti példán kívül nehéz más szigorúan stacionárius folyamatot szerkeszteni, mivel az eloszlásfüggvények azonossága eléggé szigorú feltételeket szab. Ezért az idősor analízisben általában egy gyengébb stacionaritási tulajdonságot vizsgálunk, amely csak a folyamat momentumaira fogalmaz meg feltételt.

5. Értelmezés. *Egy sztochasztikus folyamat* n**-ed rendű gyenge stacionárius folyamat**, *ha minden legfennebb* n*-ed rendű együttes momentuma létezik és független az időtengely origójának a megválasztásától, azaz időbeli eltolással értéke nem változik.*

Az értelmezés alapján egy folyamat pontosan akkor lesz **másodrendű gyenge stacionárius folyamat**, ha konstans a várható érték függvénye és a varianciafüggvénye, kovariancia- és korrelációfüggvénye pedig időbeli eltolással nem változik, azaz

(i) $E(Z_t^2) < \infty$;

(ii) $\mu_t = \mu, \sigma_t^2 = Var(Z_t) = \sigma^2$;

(iii) $\gamma(t, t+k) = \gamma_k$ $(\rho(t, t+k) = \rho_k), \forall t, k \in \mathbb{Z}$.

1. Következmény. *A 3. és 5. definíciók következménye, hogy:*

- *Egy szigorúan stacionárius folyamat, amelynek első két momentuma véges, másodrendű gyenge stacionárius folyamat is egyben (ha egy szigorúan stacionárius folyamat első két momentuma nem véges, akkor nem rendelkezik a másodrendű gyenge stacionariutás tulajdonságával).*

- *Egy másodrendű gyenge stacionárius folyamat sem feltétlenül szigorúan stacionárius folyamat.*

Néha használjuk a **kovariancia-stacionárius** vagy **gyenge értelemben stacionárius** kifejezést is a másodrendű gyenge stacionáritás kifejezésére.

6. Értelmezés. *Egy sztochasztikus folyamat* **normál-** *vagy* **Gauss-folyamat**, *ha az együttes eloszlásfüggvényei normál eloszlásúak.*

A kovariancia-stacionaritás általában sokkal gyengébb, mint a szigorú vagy akár eloszlásban való stacionáriusi tulajdonság. A normál eloszlás viszont egyértelműen meghatározott a várható értéke és a szórásnégyzete által, azaz az első két momentuma által, ezért a szigorú stacionaritás és a gyenge értelemben vett stacionaritás ekvivalens lesz Gauss-folyamatokra. Az idősoranalízis legtöbb eredménye Gauss-folyamatokra van megfogalmazva.

4.1. Az autokovariancia- és az autokorrelációs függvény

Egy $\{Z_t\}$ stacionárius folyamat esetén a várható érték függvénye $E(Z_t) = \mu$ és a varianciafüggvény $Var(Z_t) = E(Z_t - \mu)^2 = \sigma^2$ konstansok, míg a $Cov(Z_t, Z_s)$ csak a $|t - s|$ időbeli távolság függvényei. Ezért Z_t és Z_{t+k} kovarianciája

$$\gamma_k = Cov(Z_t, Z_{t+k}) = E(Z_t - \mu)(Z_{t+k} - \mu), \tag{4.2}$$

valamint Z_t és Z_{t+k} korrelációja

$$\rho_k = \frac{Cov(Z_t, Z_{t+k})}{\sqrt{Var(Z_t)}\sqrt{Var(Z_{t+k})}} = \frac{\gamma_k}{\gamma_0}. \tag{4.3}$$

7. Értelmezés. *A (4.2) összefüggéssel definiált γ_k, mint a k időbeli távolság függvénye, a $\{Z_t\}$ stacionárius folyamat* **autokovariancia-függvény***e.*

8. Értelmezés. *A (4.3) összefüggéssel megadott ρ_k a $\{Z_t\}$ stacionárius folyamat* **autokorrelációs függvény***e.*

Megjegyzés. Az autokovariancia-függvény, illetve az autokorrelációs függvény elnevezés azzal indokolható, hogy γ_k és ρ_k *ugyanazon* folyamat Z_t és Z_{t+k} valószínűségi változói közti kovariancia és korreláció. Az autokorrelációs függvényt, az angol neve alapján: ,,autocorrelation function", **ACF**-fel rövidítjük.

4.2. A parciális autokorrelációs függvény

Idősor-elemzéskor nemcsak a Z_t és Z_{t+k} közti autokorrelációra vagyunk kíváncsiak, gyakran hasznos információt szolgáltat, ha kiszámoljuk a Z_t és Z_{t+k} korrelációját miután a $Z_{t+1}, Z_{t+2}, \ldots, Z_{t+k-1}$ közbeeső valószínűségi változók lineáris komponenseit eltávolítottuk belőlük.

9. Értelmezés. *Z_t és Z_{t+k} valószínűségi változóknak a – közbeeső változók lineáris hatásának a kiküszöbölése után származó –*

$$Corr(Z_t - P_{\overline{sp}\{1,Z_{t+1},\ldots,Z_{t+k-1}\}}Z_t, Z_{t+k} - P_{\overline{sp}\{1,Z_{t+1},\ldots,Z_{t+k-1}\}}Z_{t+k})$$

korrelációját **feltételes korreláció***nak nevezzük, ahol $P_{\overline{sp}\{1,Z_{t+1},\ldots,Z_{t+k-1}\}}$ az $1, Z_{t+1}, \ldots, Z_{t+k-1}$ valószínűségi változókra kifeszített $\overline{sp}\{1, Z_{t+1}, \ldots, Z_{t+k-1}\}$ legkisebb zárt résztérre való projekció.*

A Z_t és Z_{t+k} valószínűségi változóknak a $Z_{t+1}, \ldots, Z_{t+k-1}$ változókra vonatkoztatott feltételes korrelációjára a

$$Corr(Z_t, Z_{t+k}|Z_{t+1}, Z_{t+2}, \ldots, Z_{t+k-1})$$

jelölést használjuk. Az idősor-analízisben ezt az értéket Z_t és Z_{t+k} **parciális autokorrelációjá***nak nevezzük és P_k-val jelöljük.*

10. Értelmezés. *A fent értelmezett P_k-t, mint a k függvényét, a $\{Z_t\}$ stacionárius folyamat* **parciális autokorrelációs függvényé***nek nevezzük.*

A parciális autokorrelációs függvényt, az angol neve alapján: ,,partial autocorrelation function", **PACF**-fel rövidítjük.

Megjegyzés.

1. Vegyük észre, hogy a $\{Z_t\}$ stacionárius folyamat esetén

$$Z_t - P_{\overline{sp}\{1,Z_{t+1},\ldots,Z_{t+k-1}\}}Z_t = (Z_t - \mu) - P_{\overline{sp}\{Z_{t+1}-\mu,\ldots,Z_{t+k-1}-\mu\}}Z_t, \quad \forall t \in \mathbb{Z}$$

és

$$Z_{t+k} - P_{\overline{sp}\{1,Z_{t+1},\ldots,Z_{t+k-1}\}}Z_{t+k} = (Z_{t+k} - \mu) - P_{\overline{sp}\{Z_{t+1}-\mu,\ldots,Z_{t+k-1}-\mu\}}Z_{t+k}, \quad \forall t, k \in \mathbb{Z},$$

ezért

$$\begin{aligned} P_k &= Corr(Z_t, Z_{t+k}|Z_{t+1}, Z_{t+2}, \ldots, Z_{t+k-1}) \\ &= Corr(Z_t - \mu, Z_{t+k} - \mu|Z_{t+1} - \mu, \ldots, Z_{t+k-1} - \mu) \\ &= Corr(\dot{Z}_t, \dot{Z}_{t+k}|\dot{Z}_{t+1}, \ldots, \dot{Z}_{t+k-1}), \end{aligned}$$

ahol az X valószínűségi változó centralizásából származó $X - E(X)$ valószínűségi változóra az $\dot{X}$ jelölést használtuk.

2. A parciális autokorrelációs függvényt értelmezhetjük úgy is, mint a Z_t **regressziós együtthatóját** a Z_{t+k} független és $Z_{t+k-1}, Z_{t+k-2}, \ldots, Z_t$ meghatározó valószínűségi változókra illeszkedő lineáris regressziós modellből.

4.3. ARIMA modellek

11. Értelmezés. *Ha egy $\{Z_t\}$ sztochasztikus folyamat teljesíti az*

$$\phi^p(B)(1-B)^d Z_t = \theta + \theta^q(B)a_t$$

összefüggést minden $t \in \mathbb{Z}$ időpillanatban, ahol $\phi^p(z) = 1 - \phi_1 z - \cdots - \phi_p z^p \neq 0,\ \forall |z| \leq 1$ p-ed rendű polinom, $\theta^q(z) = 1 - \theta_1 z - \cdots - \theta_q z^q \neq 0,\ \forall |z| \leq 1$ q-ad rendű polinom, θ konstans és $d > 0$ egész szám, B a $B(Z_t) = Z_{t-1}, \forall\ t \in \mathbb{N}$ szabályt követő eltolásoperátor és $\{a_t\}_{t\in\mathbb{N}}$ fehér zaj [1], akkor a $\{Z_t\}$ sztochasztikus folyamtot ARIMA(p, d, q) folyamatnak nevezzük.

Megjegyzés. $d = 0$ esetén, ahhoz hogy az ARIMA$(p, 0, q)$ folyamat megegyezzen az ARMA(p, q) folyamattal $\theta = \mu(1 - \phi_1 - \cdots - \phi_p)$-nak kell teljesülnie. Hogy az ARIMA modellt az ARMA általánosításaként tekintsük, a $d = 0$ esetén θ értékét értelmezés szerint $\mu(1 - \phi_1 - \cdots - \phi_p)$-nek választjuk.

Ha $d \neq 0$, az ARIMA(p, d, q) modell tartalmaz sztochasztikus és determinisztikus trend komponenst is: az $(1 - B)^d$ operátor szolgáltatja a sztochasztikus komponenst és a θ a determinisztikusat.

Megjegyzés. Annak ellenére, hogy az ARIMA folyamatok nem stacionáriusak, a folyamat teljes kimentelét véges számú paraméter határozza meg: $\phi_i(i = \overline{1, p})$, $\theta_j(j = \overline{1, q})$

[1] Egy $\{a_t\}$ sztochasztikus folyamat **fehér zaj**, ha azonos eloszlású, független valószínűségi változók indexelt sorozata.

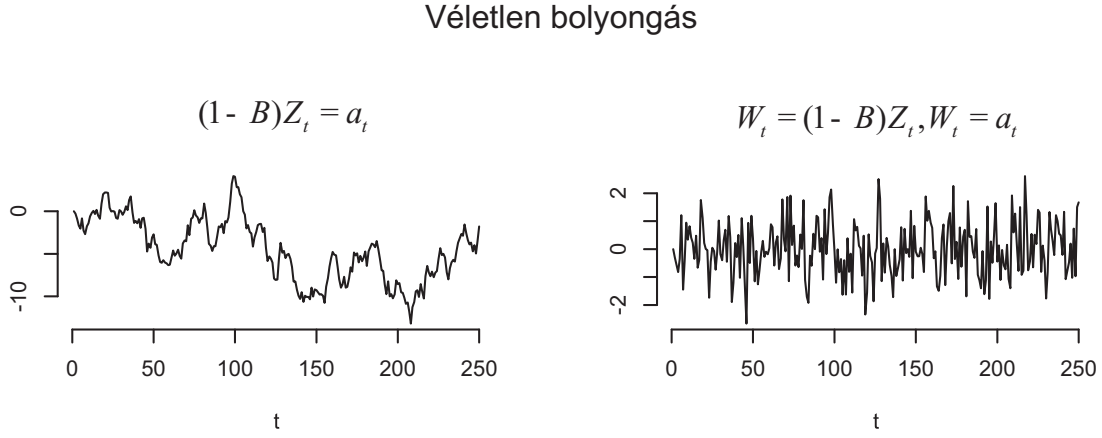
és σ_a^2. Tehát egy folyamat teljes kimenetele meghatározható, ha egy adott $Z_1, Z_2, \ldots, Z_n$ megfigyeléssorozatra illesztünk egy ARIMA modellt.

4.3.1. Véletlen bolyongás modellje

Az ARIMA(0,1,0) folyamat ismert a véletlen bolyongás modellje néven ismert.

Ha a véletlen bolyongás modellje nem tartalmaz determinisztikus trendet, azaz $\theta = 0$, akkor

$$(1 - B)Z_t = a_t.$$



4.4. A GARCH modell

A klasszikus

$$Y_t = \beta_1 X_{1,t} + \beta_2 X_{2,t} + \cdots \beta_k X_{k,t} + \varepsilon_t = X_t'\beta + \varepsilon_t \tag{4.4}$$

alakú regressziós modellhez képest (Y_t a célváltozó és $X_{1,t}, X_{2,t}, \ldots, X_{k,t}$ a magyarázó változók, a mi esetünkben az aktuális évre vonatkozó napi kivevések illetve az elmúlt k évre vonatkozó napi kivevésértékek) feltételezzük, hogy

$$\varepsilon_t = \varphi_1 \varepsilon_{t-1} + \varphi_2 \varepsilon_{t-2} \cdots \varphi_p \varepsilon_{t-p} + n_t, \tag{4.5}$$

ahol $n_t = \sigma_t \cdot e_t$ és

$$\sigma_t^2 = \theta_0 + \phi_1 \sigma_{t-1}^2 + +\phi_2 \sigma_{t-2}^2 + \cdots + \phi_r \sigma_{t-r}^2 + \theta_1 n_{t-1}^2 + \theta_2 n_{t-2}^2 + \cdots + \theta_s \sigma_{t-s}^2 \tag{4.6}$$

valamint az (e_t) változók független $N(0,1)$ eloszlású valószínűségi változók. Ezt a modellt nevezzük (r,s) paraméterű GARCH modellnek. Ez gyakorlatilag azt jelenti, hogy az σ_t^2-ekre felírt GARCH(r,s) modell ekvivalens az n_t^2-ekre felírt ARMA(m,r) modellel, ahol $m = \max\{r,s\}$. Így gyakorlatilag a következő lépések elvégzése szükséges:

1. A legkisebb négyzetek módszerével meghatározzuk a 4.4 modellből a $\beta_1, \beta_2, \ldots, \beta_p$ paraméterek becslését (ez egyúttal maximum likelihood becslés is) és ez alapján kiszámítjuk a maradékok $\hat{\varepsilon}_t$ becslését.

2. Illesztünk egy $AR(p)$ modellt az $\hat{\varepsilon}_t$ adatokra (4.5 alapján).

3. A paraméterek becsléséből kiszámítjuk az n_t értékek $\hat{n}_t$ becslését.

4. A $\hat{n}_t^2$ adatok alapján kiszámítjuk az autokorrelációs és a parciális autokorrelációs függvények becslését a

$$\hat{\rho}(\hat{n}_t^2) = \frac{\sum_{t=1}^{n-i} \left(\hat{n}_t^2 - \hat{\sigma}^2\right)\left(\hat{n}_{t+i}^2 - \hat{\sigma}^2\right)}{\sum_{t=1}^{n} \left(\hat{n}_t^2 - \hat{\sigma}^2\right)^2}, \tag{4.7}$$

ahol

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{t=1}^{n} \hat{n}_t^2.$$

Az autokorrelációs és a parciális autokorrelációs függvényekre kapott becslés segítségével azonosítható a modell rendje és így kiszámolható az előrejelzés.

Megjegyzés. A következő lépés az, hogy a GARCH modell helyett, egy olyan modellt építünk, amely nem ARMA folyamatra vezethető vissza, hanem ARIMA folyamatra (a maradékok segítségével). Ennek a tesztelését és a két modell összehasonlítását viszont csakis konkrét banki adatokon érdemes elvégezni.

Függelék

Programrészletek

```
function varargout = cash_felulet(varargin)
% CASH_FELULET M-file for cash_felulet.fig
%      CASH_FELULET, by itself, creates a new CASH_FELULET or raises the existing
%      singleton*.
%
%      H = CASH_FELULET returns the handle to a new CASH_FELULET or the handle to
%      the existing singleton*.
%
%      CASH_FELULET('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in CASH_FELULET.M with the given input arguments.
%
%      CASH_FELULET('Property','Value',...) creates a new CASH_FELULET or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before cash_felulet_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to cash_felulet_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help cash_felulet

% Last Modified by GUIDE v2.5 13-Feb-2008 12:54:49

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1; gui_State = struct('gui_Name',       mfilename,
...
```

```
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @cash_felulet_OpeningFcn, ...
                   'gui_OutputFcn',  @cash_felulet_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before cash_felulet is made visible.
function cash_felulet_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to cash_felulet (see VARARGIN)
% Choose default command line output for cash_felulet
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes cash_felulet wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = cash_felulet_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
tic; global x; global forecastv; szinek=['b' 'g' 'c' 'r'];
honapok=[0 31 28 31 30 31 30 31 31 30 31 30 31];
csucsokh=cumsum(honapok(1:12))+15; csucsok=[140 150 145 170 162 128
180 140 130 170 150 190]; M=60:100:10000;
 x=40+20*rand(4,365);
for i=1:12
    x(:,csucsokh(i))=x(:,csucsokh(i))+(2+randn(1,1))*csucsok(i)/4;
end forecastv=mean(x(1:3,:))+3*std(x(1:3,:)); figelore=figure; for
j=1:4
  subplot(2,3,j);
plot(x(j,:),szinek(j));hold on;
end; subplot(2,3,5); plot(forecastv);subplot(2,3,6);
plot(abs((forecastv-x(4,:))./x(4,:)));
ir= str2double(get(handles.edit2,'String')); T=
str2double(get(handles.edit3,'String'));
for k=1:length(M) sv=M(k); HC(k)=0;ki=0; for i=1:364
    HC(k)=HC(k)+sv*ir/365;
    sv=sv-x(4,i);
    if sv<forecastv(1,i+1)
       ki=ki+1; kazetta(1,ki,k)=sv;   sv=M(k); HC(k)=HC(k)+T;
    end
end
end
global m; [m,h]=min(HC); global mm; mm=M(h); fig0=figure;
subplot(2,1,1); [mi,ho]=min(kazetta(1,:,h));
bar(kazetta(1,1:ho-1,h)); subplot(2,1,2);
bar(kazetta(1,1:ho-1,h)/M(h)*100);
hat=sum(kazetta(1,1:ho-1,h))/((ho-1)*M(h)); fig1=handles.axes1;
axes(fig1); cla(fig1); plot(M,HC);hold on; plot([M(h) M(h) 0],[0 HC(h)
HC(h)],'r'); sv(1)=M(h);ir=0.08; HC(h)=0;ki=0;T=5; global utem;
utem=zeros(1,365); for i=1:364
    HC(h)=HC(h)+sv(i)*ir/365;
    sv(i+1)=sv(i)-x(4,i);
    if sv(i+1)<forecastv(1,i+1)
       ki=ki+1; kazetta(1,ki,k)=sv(i+1);   sv(i+1)=M(h); HC(h)=HC(h)+T;
```

```
        utem(1,i+1)=M(h);
    end
end fig2=handles.axes2; axes(fig2); plot(utem); fig3=handles.axes3;
axes(fig3); plot(sv); t1=toc

function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double

% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
tic; keresztezodesi_sz= str2double(get(handles.edit4,'String'));
mutaciok_sz= str2double(get(handles.edit5,'String')); po=
str2double(get(handles.edit6,'String')); iter=
str2double(get(handles.edit7,'String')); ir=
str2double(get(handles.edit2,'String')); szall=
str2double(get(handles.edit3,'String')); global x; global forecastv;
napok_sz=365; global utem; global mm; atlag=mm; szoras=mm*0.1;
kezdeti_keszlet=mm;
kife=forecastv;%az elorejelzett kifizetesek
kifv=x(4,:); k_pop=general_kezdeti(po,napok_sz,mm,utem); [koltseg
k_pop keszletv
kazettaki]=kiert(k_pop,kifv,kife,ir/365,kezdeti_keszlet,szall);
pop=k_pop;sugar=3*szoras; for i=1:iter
    kik_mutalnak=floor(1+po*rand(1,floor(mutaciok_sz*po)));
    sugar=sugar/i;
    for j=1:length(kik_mutalnak)
        uj_pop1(j,:)=mutal(pop(kik_mutalnak(1,j),:),kifv,floor((iter-i)/iter*20),sugar);
    end
    kik_kereszt=floor(1+po*rand(2,floor(keresztezodesi_sz*po/2)));
     for j=1:floor(keresztezodesi_sz*po/2)
         uj_pop2(j,:)=keresztez(pop(kik_kereszt(1,j),:),pop(kik_kereszt(2,j),:));
     end
        %kik_mutalnak=floor(1+mekk(1,1)*rand(1,floor(mutaciok_sz*mekk(1,1))));
    %for j=1:length(kik_mutalnak)
    %    uj_pop1(j,:)=maxok(uj_pop(j,:),30);
    %end
    uj_pop=[pop; uj_pop1;uj_pop2];
    [koltsegek uj_pop keszletv kazettaki]=kiert(uj_pop,kifv,kife,ir/365,kezdeti_keszlet,szall);
```

```
    mekk=size(uj_pop);
    rendezni=[koltsegek uj_pop (1:mekk(1,1))'];
    rendezett=sortrows(rendezni,1);
    pop=[rendezett(1:po-20,2:napok_sz+1); pop(1:20,:)];
    HC2(1,i)=rendezett(1,1);
%    HCbuntet(1,i)=kiert(pop(1,:),kifv,ir,kezdeti_keszlet,szall);
if i==iter
    disp('A minimlis kltsg rtke: ');
    disp(min(HC2));
end
end
fig4=handles.axes4; axes(fig4); plot(HC2,'r');
fig45=figure;
%subplot(2,1,1);
plot(HC2,'r');
%subplot(2,1,2);
%plot(HCbuntet,'r');
fig5=handles.axes5; axes(fig5); plot(pop(1,:));
fig55=figure; bar(kazettaki(rendezett(1,napok_sz+2),:));
fig6=handles.axes6; axes(fig6);
plot(keszletv(rendezett(1,napok_sz+2),:));
t2=toc

function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double

% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
```

```
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a double

% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6 as a double

% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a double


% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
display Viszlt close(handles.figure1); close all;


function mutans=mutal(v,kif,lepessz,r);
    global mm;
    v(1,1)=v(1,1)+mm;
    vv=cumsum(v);
```

```
    kiff=cumsum(kif);
    kiurul=kiff-vv;
    for i=1:length(v)
        if kiurul(i)<0
            v(1,i)=mm;
            kiurul=kiurul+mm;
        end
    end
v1=maxok(v,30);
for i=1:length(v)
leptet=floor(3*rand(1,1)-1);
        mennyit=floor((20-lepessz)*rand(1,1));
        if (i+leptet*mennyit>0)&(i+leptet*mennyit<length(v)+1)
        seged=v1(i);
            v1(i)=v1(i+leptet*mennyit);
            v1(i+leptet*mennyit)=abs(seged+floor(2*r*rand(1,1)-r)*abs(leptet));
        elseif (i-leptet*mennyit>0)&(i-leptet*mennyit<length(v1))
        seged=v1(i);
            v1(i)=v1(i-leptet*mennyit);
            v1(i-leptet*mennyit)=abs(seged+floor(2*r*rand(1,1)-r)*abs(leptet));
        end
end
mutans=v1;


function mvekt=maxok(v,h)
l=length(v);
rendezni=[v' (1:l)'];
rendezett=sortrows(rendezni,1)';
indexek=rendezett(2,l-h+1:l);
v1=zeros(1,l);
for i=1:length(indexek)
    v1(1,indexek(i))=v(1,indexek(i));
end
mvekt=v1;


function kif=kifizetesek(napok,atlag,szoras)
honapok=[0 31 28 31 30 31 30 31 31 30 31 30 31];
csucsokh=cumsum(honapok(1:12))+15;
```

```
csucsok=[140 150 145 170 162 128 180 140 130 170 150 190];
kif=floor(rand(1,napok)*atlag/10+3*szoras*rand(1,napok));
for i=1:length(csucsok)
    kif(1,1+floor(csucsokh(i)*napok/365))= kif(1,1+floor(csucsokh(i)*napok/365))+atlag/3+szoras*ra
end


function kolt=kiertsima(pop,kifizet,ir,kk,transp)
mer=size(pop);
kolt1=kk*ir*ones(mer(1,1),1);
keszlet=kk*ones(mer(1,1),1);
for j=1:mer(1,1)
for i=1:length(kifizet)
    if pop(j,i)~=0
           kolt1(j,1)=kolt1(j,1)+transp;
    end
    if keszlet(j)-kifizet(i)>0
         kolt1(j,1)=kolt1(j,1)+(keszlet(j,1)-kifizet(1,i)+pop(j,i))*ir;
    %else kolt1(j,1)=kolt1(j,1)+abs(keszlet(j,1)-kifizet(1,i))^2*ir;%buntetofuggveny
    end
        if pop(j,i)~=0
            keszlet(j,1)=pop(j,i);
        end
end
end
kolt=kolt1;


function [kolt,pop1,keszlet,kazetta]=kiert(pop,kifizet,kifizete,ir,kk,transp)
pop1=pop;
felsohatar=10000;
mer=size(pop);
kolt1=kk*ir*ones(mer(1,1),1);
keszlet=[kk*ones(mer(1,1),1) zeros(mer(1,1),mer(1,2))];
kazetta=zeros(mer(1,1),mer(1,2));
for j=1:mer(1,1)
for i=1:length(kifizet)-1
     if keszlet(j,i)-kifizete(i)>0
        pop1(j,i)=0;
         kolt1(j,1)=kolt1(j,1)+(keszlet(j,1)-kifizet(1,i))*ir;
```

```
          keszlet(j,i+1)=keszlet(j,i)-kifizet(i);
    else %prompt szallitas
       kolt1(j,1)=kolt1(j,1)+transp;
       kazetta(j,i)=keszlet(j,i);
       kul=cumsum(pop1(j,:)-kifizet);
        m=1;
        for hh=i:length(kifizet)
            if kul(1,hh)<=0
                m=m+1;
            else m=m;
            end
        end
                if i+m-1<366
              kazetta(j,i)=keszlet(j,i);
              keszlet(j,i)=min(max(keszlet(j,i)+abs(kul(1,i+m-1)),kk),felsohatar);
              pop1(j,i)=min(max(keszlet(j,i)+abs(kul(1,i+m-1)),kk),felsohatar);
              keszlet(j,i+1)=keszlet(j,i)-kifizet(i);
                else
              kazetta(j,i)=keszlet(j,i);
              keszlet(j,i)=min(max(keszlet(j,i)+abs(kul(1,365)),kk),felsohatar);
              pop1(j,i)=min(max(keszlet(j,i)+abs(kul(1,365)),kk),felsohatar);
              keszlet(j,i+1)=keszlet(j,i)-kifizet(i);
                end
     end
    end
end
kolt=kolt1;


function utod=keresztez(u,v)
p=rand(1,length(u));
for i=1:length(u)
    if p(1,i)<0.5
        utod(1,i)=u(1,i);
    else
        utod(1,i)=v(1,i);
    end
end
```

Irodalomjegyzék

[1] Anderson, O. D. (1976). *Time Series analysis and forecasting: The Box-Jenkins approach.* Butterworths, Series R.

[2] Box, G. és Jerkins, G. (1976). *Time series analysis: forecasting and control.* Holden-Day, San Francisco.

[3] Brockwell, Peter J. and Davis, Richard A. (1987). *Time series: theory and methods.* Springer, New York.

[4] Brockwell, Peter J. and Davis, Richard A. (2002). *Introduction to time series and forecasting.* Springer.

[5] Chatfield, Chris. (1985). *The analysis of time series: an introduction.* Chapman and Hall, New York.

[6] Diggle, P. J. (1990). *Time series: A biostatistical introduction.* Oxford.

[7] Fuller, W. A. (1996). *Introduction to statistical time series.* Wiley Series in Probability and Statistics. John Wiley & Sons Inc., New York.

[8] Hamilton, James D. (1994). *Times series analysis.* Princeton University, Princeton.

[9] Kippers, Jeanine (2004). *Empirical Studies on Cash Payments.* ERIM, Erasmus University Rotterdam.

[10] Michelberger, Pál; Szeidl, László és Várlaki, László. (2001). *Alkalmazott folyamatstatisztika és idősoranalízis.* Typotex, Budapest.

[11] Pankratz, A. (1983). *Forecasting with univariate Box-Jerkins Models.* John Wiley & Sons, New York.

[12] Pankratz, A. (1991). *Forecasting with dynamic regression models.* John Wiley & Sons, New York.

[13] Soós, Anna (2001). *A valószínűségszámítás elemei. I. kötet.* Egyetemi Kiadó, Kolozsvár.

[14] Tong, H. (1996). *Non-linear time series.* Clarendon Press, Oxford.

[15] Tukey, John (1977). *Exploratory data analysis.* Addison Wesley.

[16] Wei, William W. S. (2006). *Time series analysis.* Addison Wesley, Reading, Massachusett.