ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

HABILITÁCIÓS TÉZISFÜZET

DR. SCHUSTER GYÖRGY (DOCENS)

Idősorok alkalmazása szoftverminőség előrejelzésére

BIZTONSÁGTUDOMÁNYI DOKTORI ISKOLA

Tartalomjegyzék

Motiváció

A szoftver kritikus sikertényező[1].

Manapság gyakorlatilag nincs olyan kicsit összetettebb műszaki alkotás, amelyben nincs szoftver támogatás.

A szoftver elemek megjelennek minden területen a szórakoztató rendszerektől az intelligens forgalomirányításokon keresztül a kritikus fedélzeti rendszerekig.

Minél több biztonságkritikus feladatot bízunk ezekre a fedélzeti rendszerekre, annál kevesebb feladat hárul az irányítási folyamatokban résztvevő emberekre, továbbá olyan funkciókat is meg tudunk valósítani, amelyek korábban elképzelhetetlenek voltak.

Ez viszont azt jelenti, hogy a rendszerek növekvő bonyolultsága növeli a hibák előfordulásának lehetőségét[2].

Sok rendszer esetén egy szoftverhiba komoly problémát jelenthet, de megoldható, hogy a kérdéses rendszer leállítható, vagy valamilyen biztonsági állapotba hozható.

Erre jó példa egy autó motorirányító szoftvere, meghibásodás esetén - ez lehet hardver hiba is - a jármű leállhat az út szélén és várhatja a javítást. A megbízhatósági szintnek itt is nagyon magasnak kell lennie, de ez a lehetőség azért fennáll[1]

Azonban vannak olyan rendszerek, ahol a meghibásodás katasztrofális következményekkel járhat.

Tekintsünk egy személyszállító automatizált légijárművet. Itt akár egy kritikus hardver vagy szoftver meghibásodás katasztrofális lehet.

Erre jó példa a Aurora Flight Sciences „Pegasus” utasszállító légijármű (PAV) június 4-i lezuhanása[6].

Az esemény oka egy mechanikai meghibásodásból eredő hibás szoftver „döntés”, amely a levegőben leállította az emelő motorokat. Itt csak azért nem voltak áldozatok, mert a kérdéses teszt repülés folyamán nem voltak emberek a fedélzeten.

Lényeges kiemelni, ha a fedélzeti szoftver megfelelően működik, akkor az esemény egy egyszerű mechanikai meghibásodás lett volna, és a repülőeszköz nem kerül „nem javítható” státuszba.

Itt a fedélzeti rendszer nem kerülhet olyan állapotba, amely az irányított rendszer leállásához vezethet[2]

[1] Ezeket a rendszereket a sokszor úgynevezett működésbiztos rendszereknek nevezzük[3].
[2] Az ilyen rendszereket veszélybiztos rendszereknek nevezzük[3].

A magyar terminológia az ilyen besorolású szoftvereket „biztonság kritikus” szoftvereknek nevezi. Azonban sajnos itt nem tesz különbséget a működés biztonság és az információ biztonság között[3].

Kutatási területem a működés biztonság témaköre, egyáltalán nem érinti az információ biztonság kérdéseit.

A feltett kérdés, amelyre a kutatás irányul a következő:

Egy még meg nem írt szoftver minőségét milyen módon tudjuk megbecsülni?

Tekintsük a következő szituációt: adott egy biztonságkritikus fejlesztési igény és adott három szoftvercég, amelyek azonos jogosításokkal és minősítéssel, illetve besorolással rendelkeznek.

Ilyen minősítés lehet például a CMM (Capability Maturity Model)[8], vagy a CMMI (Capability Maturity Model Integration)[9].

Amennyiben a kérdéses gyártóknak ezen minősítései érvényesek, a megrendelőknek nincs okuk a kétkedésre. Ekkor a megrendelő valamilyen másodlagos szempont szerint választ gyártót. Ez nem mindig a legjobb módszer.

Más mérnöki alkalmazásokban az alkatrészek statisztikai adatai lehetővé teszik az elkészített berendezések megbízhatóságának meglehetősen pontos becslését. Sajnos ez szoftverek esetében ilyen módon nem működik. Viszont rendkívül fontos lenne.

Ez a gondolat nem új, 1972-ben Michael Fagan felvetette ennek szükségességét, amiből a szoftver inspekció fejlődött ki[10]. Habár az szoftver inspekció sokat javított a szoftvereken[11] eltért az eredeti statisztikai koncepciótól.

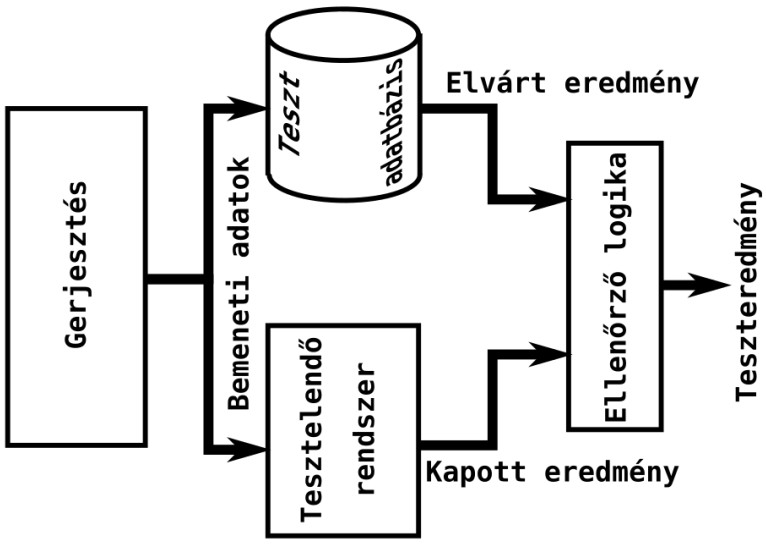
Kutatásom célja, hogy olyan statisztikai módszert, vagy módszerek kombinációját válasszuk ki, amelyek előrejelezhetik a még el sem készült szoftver várható minőségét.

A probléma

Bár az előző fejezetben említettem, hogy statisztikai módszert, vagy módszereket keresek a minőség előrejelzésére, felmerül a kérdés, hogy az elkészült szoftvert a tesztelés során hibamentessé lehet-e tenni. Ha igen, akkor nincs szükség az előzetes vizsgálatra, de:

- egy relatív nagy méretű szoftver tesztelése, ha az sok hibát tartalmaz, igen hosszadalmas és rengeteg időt felemésztő tevékenység,
- a szoftverek elfogadható idő alatt teljes mértékben nem tesztelhetők.

[3] Az angol terminológia az információ biztonságra a „Security Critical” kifejezést, míg a működés biztonságra a „Safety Critical” kifejezést használja



1. ábra: A tesztelés válzata.

A második pontot tételként is kimondom és bizonyítom.

Tétel 1. [4] A szoftverek elfogadható idő alatt teljes mértékben nem tesztelhetők.

Bizonyítás. [4] Tegyük fel, hogy van egy olyan C nyelven megírt programunk, amely három **`short int`** típusú változót ad össze. Ezt a programot kívánjuk teljes mértékben tesztelni.

`short int` változó típus 16 bites.

Mivel minden bemeneti lehetőséget meg kell vizsgálnunk, így a bemeneti lehetőségek száma 2^{48}. Ez a számérték $281\ 474\ 976\ 710\ 656$

Ha feltételezzük, hogy olyan teszterünk van, amely másodpercenként 10^9 tesztet tud elvégezni, akkor kerekítve $281\ 475$ másodpercre van szükségünk a teljes teszteléshez. Egy napban 86400 másodperc van, így kicsit több, mint 3 nap és 6 óra a tesztelés ideje.

A teszteléshez szükséges még egy teszt adatbázis is, ennek mérete 768 TB.

A tesztelés vázlata a 1. ábrán látható.

A változók számának növekedésével a tesztelési idő és a teszt adatbázis mérete exponenciálisan növekszik.

A problémát az állapotgépek hatványozottan bonyolítják. Ez alapján kijelenthetjük, hogy egy közepes, vagy nagy bonyolultságú szoftver esetén a teljes tesztelés gyakorlatilag lehetetlen.

□

Definíciók és megszorítások

A hipotézisek kimondása előtt néhány definíciót célszerű megadni és a vizsgálat körülményeit le kell rögzíteni.

Definíciók

Definíció 1. Szoftver: a programok és az adatok összessége[5].

Megjegyzés: Egy berendezéshez nem csak a programok szükségesek, hanem a konfigurációs adatok is. Egy hibátlan program is képes hibásan működni, ha a felhasznált adatok nem megfelelőek. Erre jó példa, a hibás konfigurációs adatok megadása.

Definíció 2. A szoftver megbízhatóság: egy adott, meghatározott időtartamon belül, adott környezetben a kérdéses szoftver hibamentes működésének valószínűsége[5].

Definíció 3. Eltérés: (failure) a felhasználó által észlelt váratlan szoftver viselkedés[5].

Definíció 4. Hiba: (fault) olyan eltérés által okozott szoftver jellemző, amelynek hatása van a szoftver működésére[5].

Megjegyzés: A vizsgálatban az eltérést is hibaként kezelem. Ennek oka az, hogy az eltérés is egy nem megfelelőség a specifikációt tekintve.

A vizsgálat területei

A vizsgálatot két területen tudjuk végezni, ezek:

1. a már elkészült szoftver meghibásodási - javítási idősorát vizsgáljuk,
2. a fejlesztés során vizsgáljuk a fejlesztési lépésekben a hibák előfordulásának idősorait.

Mindkét módszer ígéretesnek látszik, azonban az első pontban megemlített módszerrel kapcsolatban felmerültek problémák, nevezetesen:

- ha az elkészült szoftver általános felhasználású, akkor a hibaesemények gyűjtése nehézkes. Igaz, hogy az ilyen jellegű szoftverek ugyan szolgáltatás ként felkérik a felhasználókat, hogy a hibaüzeneteket küldjék el a gyártóknak, de sajnos a felhasználási időt ezek a beérkező üzenetek nem tartalmazzák. Az általános célú szoftverek túlnyomó részt nem biztonság kritikusak,

- amennyiben a szoftver általános felhasználású, de biztonságkritikus, mint például egy gépjármű fedélzeti beágyazott rendszerének szoftvere, az információk begyűjtése szintén nehézkes, kiemelten akkor, ha a meghibásodás nem kritikus,
- egyedi biztonságkritikus alkalmazásokban a hiba előfordulás annyira ritka lehet, hogy statisztikai vizsgálatra nem alkalmazható.

Ezért a második, a fejlesztés idejű vizsgálatokat részesítettem előnyben.

A vizsgálat megszorításai

A fejlesztés során a fejlesztőknek és a fejlesztési folyamatnak a következő feltételeket kell teljesíteni:

1. csak azt a fejlesztési folyamatot vizsgáljuk, ahol a fejlesztők már kipróbált és tesztelt szoftver elemeket használnak,
2. olyan fejlesztési folyamatot vizsgálunk, amelynek az eredménye valamilyen biztonságkritikus termék,
3. nem vizsgálható olyan folyamat, amelyben junior alkalmazottak és gyakornokok dolgoznak. Amennyiben van valamilyen minősítési rendszer a résztvevőkkel szemben, akkor a vizsgált projektben, vagy projektekben csak minősített alkalmazottak vehetnek részt,
4. ha valamilyen felhasznált program elemben, ez lehet akár belső, akár külső hiba van, amely meghibásodást eredményez, azt fejlesztési hibának vesszük.

   Ennek indoklása az, hogy ez működési meghibásodást okozhat.

Problémát okozhat az is, hogy egy fejlesztő cég számos architektúrán dolgozhat, számos fejlesztői környezetet használva. Ezeket a vizsgálat során célszerű felhasználni, de így ezért nehezen hasonlíthatók össze. Erre ad megoldást a 1. hipotézis, illetve a bizonyított 1. tézis.

1. Hipotézisek

Hipotézis 1. Az időbeli vizsgálat helyett a végrehajtott gépi utasítások számával skálázzuk az idősort.

Az eltérő processzor architektúrák és fejlesztői környezetek közti különbséget megfelelő arányossági tényezővel történő beszorzással egyenlíthetjük ki.

Hipotézis 2. Két gyártó közötti különbséget úgy határozhatjuk meg abban az esetben, ha az idősoraik átlaga közel azonos, hogy az adott idősorok trendjét vizsgáljuk meg.

A trendvizsgálat eszköze a Hurst-analaízis. A trend jellegét a lineáris regressziós egyenes (1. fokú regressziós polinom) megadja.

Hipotézis 3. A gyártó minőségi javulásra fordított erőfeszítései megbecsülhetőek az idősor autokorrelációs függvényének alakulásával.

Hipotézis 4. Az idősor alakulását a negyed és ötödfokú regressziós polinom mutatja be kellően áttekinthetően.

2. Kutatási módszertan

A kutatás során több módszert alkalmaztam. Ezek a módszerek a következők voltak:

1. olyan cégek adatainak felhasználása, amelyek biztonságkritikus fejlesztéssel foglalkoznak,
2. szimulált adatok vizsgálata a megfelelő regressziós közelítés és szint kiválasztására.

Az első módszernél komoly akadályba futottam, személyes kapcsolataimat kihasználva rákérdeztem - nagyrészt középvezetői szinten -, hogy adnának-e ilyen jellegű adatokat?

Kétféle választ kaptam:

- ilyen adatokat nem gyűjtenek,
- az ilyen jellegű adatok vállalati titkot képeznek, így nem kiadhatók.

Egyetlen cégtől kaptam felhasználható és megfelelő adatokat. Ezeket az adatokat használni fogom a vizsgálatoknál.

Sem az adatot megadó, sem az elutasító cégek a dolgozatban nem lesznek nevesítve.

Mivel így igen kevés adat állna rendelkezésemre, ezért más megoldást kellett találnom. A hallgatói tevékenységet vizsgáltam két egymást követő szemeszterben.

Ez az adatgyűjtés teljesen név nélkül és nem nyilvánosan történt. Végül két hallgatót választottam ki, akiknek a következő kritériumoknak kellett megfelelnie:

- a hallgatónak a tantárgyakat elsőre kell teljesítenie,
- a hallgatónak minden egyes laboratóriumi gyakorlatot teljesítenie kell, nem hiányozhat,

- a hallgatónak minden gyakorlati zárthelyi dolgozatát legalább elégségesre teljesítenie kell, nem ismételhet zárthelyit,
- a hallgatónak a két félévét azonos tantervben kell teljesítenie,
- a hallgatónak a laborvezetője ugyanannak az oktatónak kell lennie.

A hallgatóknál eltértem az intervallumszámítás módszerétől. Ennek oka az, hogy a hallgatók kezdőnek tekinthetők programozási szempontból, így rájuk nem a szemantikai hibák jellemzőek, hanem túlnyomó részt szintaktikai hibákat vétenek.

Ezért az intervallum becslése nem a futtatott kód hosszából, hanem a fordított kód méretéből történt.

A két vizsgált félévben a hallgatók kétszer 12 laboratóriumi gyakorlaton vesznek részt a zárthelyiket is figyelembe véve.

Laboratóriumi gyakorlatonként az átlagosan megírt kódok száma három - négy. A tapasztalatok szerint programonként átlagosan 9 - 10 szintaktikai hibát követnek el.

Az intervallumot önkényesen úgy határoztam meg, hogy a maximális hibaszám a kiválasztott intervallumban 10 körüli legyen. Így 100 intervallumot tudtam elkülöníteni.

Ez alapján le lehetett vonni következtetéseket, bár a vizsgált minta viszonylag kicsi volt.

3. Hipotézisek feldolgozása

3.1. Az idősorok skálázása végrehajtott gépi utasításokkal

Az 1 hipotézis alkalmazásának indoklása a következő. Akár futásidőben, akár fejlesztési időben[4] a szoftver futásideje nem mérhető exakt módon. Továbbá eltérő architektúrák és fejlesztői környezetek esetén az idő nem összehasonlítható.

Vegyük példának a következő esetet! A fejlesztő cég két különböző architektúrát használ, legyen az egyik egy ARM Cortex R5 magos eszköz, a másik egy AVR8 architektúra.

Tegyük fel, hogy az ARM processzor 140MHz, míg az AVR8 16MHz órajel frekvenciával működik. Ha azonos „feladatot” adunk a két processzornak, a végrehajtási idő jelentősen változik.

Első közelítő számításunk szerint az ARM 875%-al gyorsabban fogja elvégezni a kérdéses feladatot. A valóság az, hogy ennél sokkal gyorsabb lesz. Ennek oka az eltérő utasításkészlet és az eltérő felépítés.

Tehát a végrehajtott utasításokra vetített idősor sokkal jobban kifejezi az idősor jellegét.

[4] Az idő itt időszakot jelent.

Az utasításkészlet eltérésének torzítása arányossági tényezőkkel kiküszöbölhető. Vegyük a következő példát!

Tegyük fel, hogy egy adott feladatot az ARM 1000 utasítással old meg, az AVR8 1200 utasítással[5]. Ekkor a az összehasonlítás érdekében az ARM által végrehajtott utasítás számot 1,2-el meg kell szoroznunk.

Megjegyzés: Biztonságkritikus fejlesztésekben kizárólag C programnyelven írt programokkal foglalkoztam. Amennyiben a más programnyelvek használata is szóba kerül, az további vizsgálatokat feltételez.

Amennyiben több architektúrával dolgozunk, célszerű kiválasztani azt, amelynek fordítóprogramja a legtöbb gépi utasításra fordít és ezt venni alapul.

A következő kérdés, hogy hogyan tudjuk a megfelelő szorzószámot meghatározni. Egy sima assembly kód nem a legjobb megoldás. Erre a célra a „Dhrystone benchmark” teljesítmény tesztet vettem figyelembe[14].

Az általam kiválasztott vizsgálati programok:

- nem javított buborékrendezés véletlenszerűen generált 256 elemen 100 000 futtatás,
- SHA-1 hash[15] algoritmus véletlenszerűen generált 64 bájton 100 000 futtatás,
- AES-128 kódoló és dekódoló[16] algoritmus véletlenszerűen generált 7 bájton, véletlenszerűen generált 7 bájtos kulccsal, a futtatások száma 1000x1000,
- RSA titkosítás kódoló és dekódoló[17] algoritmus véletlenszerűen generált 4 bájton 64 kulcspárral, a futtatások száma 1000x64.

A fenti algoritmusok nem használják ki a specifikus hardver eszközöket, tehát nem végeznek I/O műveleteket és nem végeznek lebegőpontos számításokat sem, továbbá nem használnak hardveres osztást. Ezt azért nem vettem figyelembe, mert ez a a hardveres gyorsítás nem áll rendelkezésre minden összehasonlítandó architektúrák esetén.

Nyilvánvaló tény, hogy a biztonságkritikus szoftver alkalmazások nagyrészt I/O specifikusak, az összehasonlító vizsgálatba használatuk nem célszerű, de felhasználásban az ilyen utasítások közönséges utasításként szerepelnek eltekintve a hardver gyorsítások alkalmazásától.

Az előzőekben említett hardveres osztás utasítást egyetlen utasításként kezeltem.

A végrehajtott gépi utasításokban történő skálázás esetén óriási számokat kapunk egy intervallumra, ennek oka az, hogy két hiba között igen sok utasítás kerül végrehajtásra.

Vegyünk példának egy 100MHz órajel frekvenciával működő valós RISK architekturás processzort.

[5]Ez egy hipotetikus példa, ennél sokkal nagyobb programokat készítünk.

Ez azt jelenti, hogy egy másodperc alatt 10^8 utasítást hajt végre.

Tegyük fel, hogy folyamatos futásnál óránként jelentkezik egy hiba és az alapintervallumot úgy határozzuk meg, hogy az a leggyakoribb hiba előfordulásnál 24 hiba legyen, akkor ez folyamatos futásnál 24 órát jelent.

Mivel feltételeztük, hogy valós RISK architektúrájú processzorunk van, így minden egyes órajel egy utasítást jelent (ami a valóságban nem így van), ez $86400\ 10^8 = 8.64\ 10^{12}$ utasítást jelent.

Megjegyzés: A kísérleti futtatások során többen feltették a kérdést, hogy miért nem használtam az aszimptotikus futásidő analízist.

A válasz az, hogy az úgynevezett nagy-ordó $(O(\dots))$ aszimptotikus függvény jöhetett volna szóba. Ez alkalmas algoritmusok összehasonlítására, de nem alkalmas két architektúra assembly szintű programfuttatásának összehasonlítására.

Megjegyzés: A másik tipikus kérdés az volt, hogyan biztosítottam azt, hogy a két vizsgált hardver ugyanazokat véletlenszerűen generált bemeneteket futtatta.

A válasz az, hogy a véletlen adatokat nem a célhardver generálta, hanem ezt a feladatot PC-n egy Python program végezte.

Megjegyzés: A harmadik gyakori kérdés az volt, hogy az I/O utasításokat nem vizsgáltam, akkor hogyan kerültek az adatok a PC-ről a célhardverre.

Az adatátvitel módja USB-soros kommunikáció volt (mivel a 8 bites eszköz nem volt alkalmas hálózati kommunikációra), az aktuális próbafuttatás az átvitel után indult el. A teszt adatok szintén USB-soros kommunikációs csatornán kerültek vissza a Python programba.

Megjegyzés: A következő kérdés az, hogy miért gépi utasításokban számolok és miért nem órajelben.

Ez teljesen jogos, mert órajelben számolni sokkal könnyebb, és ha csak RISK architektúrákban gondolkodunk, akkor jó közelítéssel helyes is.

Azonban CISK esetben már az órajel nem megfelelő és ekkor még ott van a hardveres osztás utasítás kérdése is, ami változó órajel számmal hajtódik végre.

3.2. Trendvizsgálat

A fejlesztési folyamatban a hibák előfordulását mérjük. Ez lehet bármilyen hiba, de cégek esetén ezek az adatok nagyrészt a tesztelési folyamatokból származó értékek[6].

[6]Lényeges, hogy bármilyen tesztelési folyamatból

A felderített és rögzített hibákból és a 3.1 fejezetben ismertetett végrehajtott utasítás számból idősort képezünk és ennek a viselkedését fogjuk vizsgálni.

A trendvizsgálat módja a Hurst-analízis, amely az úgynevezett Hurst-exponensre épül.

A Hurst-exponens egy olyan matematikai mutató, amelyet a Hurst-függvény segítségével számítanak ki. Walter R. Hurst, angol hidrológus nevéhez köthető, aki az 1950-es években kutatásokat végzett az árvizek várható időtartamának és intenzitásának előrejelzésével kapcsolatban.

A Hurst-exponens az idősorokban, vagy adatsorokban található önkülönbözőséget, vagy hosszú távú függőség mértékét méri.

Az önkülönbözőség azt jelenti, hogy az adatsorok korábbi eseményei milyen hatással lehetnek a jövőbeli eseményekre[19].

Ezen túlmenően a Hurst-exponens értéke az adatsor fraktál jellegét is megmutatja [20]. Ezt a lehetőséget a vizsgálatban nem használtam ki.

A Hurst exponens számításának lépései:

1. számítsuk ki az eredeti adatsor átlagát és vonjuk ki minden egyes értékből (az adatsort normáljuk),

$$v_{in} = v_i - \frac{1}{n}\sum_{j=0}^{n-1} v_j, \quad i = \{0, \ldots, n-1\} \quad \text{ahol:}$$

   v_i az eredeti adatsor eleme,

   v_{in} a normált adatsor eleme.

2. a normált adatsort bontsuk rész adatsorokra, rendre felére, negyedére, nyolcadára és így tovább,

3. számítsuk ki ezeknek a rész adatsoroknak az átlagát,

$$a_k = \frac{1}{m}\sum_{l=0}^{m-1} v_l, \quad l = \{0, \ldots, m-1\} \quad \text{ahol:}$$

   a_k a részintervallum átlaga,

   m a részintervallum elemeinek száma.

   az egyszerűség kedvéért a dolgozatban a részintervallumokat újra indexeltem, a programban természetesen nem,

4. az előző részintervallumok átlagait vonjuk ki az eredeti normált adatsor elemeiből, minden elemből, minden átlagot.

   Jelöljük az így kapott eltéréseket d_o-nak, ahol: $o = \{0, \ldots, k\, n - 1,$

5. számítsuk ki az eltérések szórását,

$$S = \sqrt{\frac{1}{k\,n-1}\sum_{o=0}^{k\,n-1}(d_o - R)^2}, \quad \text{ahol:}$$

S az eltérések módosított szórása,

R az eltérések átlaga

6. a Hurst-exponens a következő:

$$\frac{R}{S} = N^H, \quad \Rightarrow \quad H = \frac{\log \frac{R}{S}}{\log N} \quad \text{ahol:}$$

H a Hurst-exponens.

A Hurst-exponens jelentése[21]:

$0 \leq H < 0.5$ ebben az esetben az idősor nem trendtartó,

$H = 0.5$ ebben az esetben az idősor teljesen véletlenszerű,

$0.5 < H \leq 1$ ebben az esetben az idősor trendtartó, minél magasabb az érték, az idősor annál inkább trendtartó.

A Hurst-exponens számításának számos módszere létezik, én a legegyszerűbbet használom és a vizsgálatok alapján ez jól működik.

Szimulációs vizsgálatok:

A Hurst-exponens ellenőrzését néhány szimulált és egy valós esettel, illetve a hallgatói adatsorok vizsgálatával végeztem el.

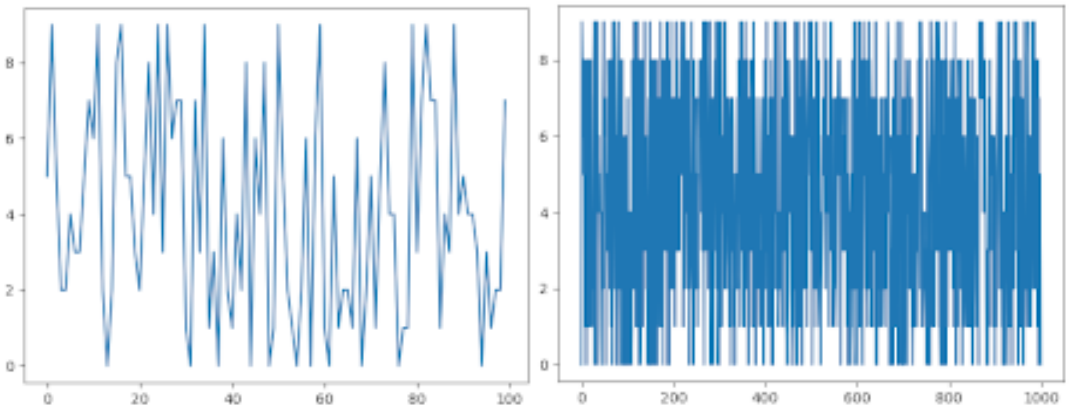
A szimulált idősorok azonos jelleggel 100 és 1000 eleműek. A megszorítás minden egyes idősorban az, hogy a mintában nem lehet negatív elem, hiszen negatív hibaszám nem lehetséges.

1. Az idősor teljesen véletlenszerű. A véletlen komponens értéke 0 és 10 között mozog.

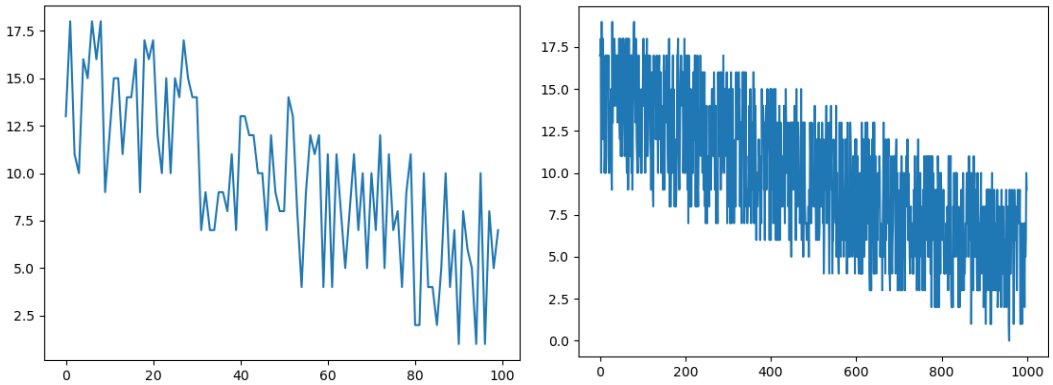
   Látható, hogy a két minta között az eltérés minimális és mind a kettő a 0.5-ös érték közelében van.

2. Az idősor egy lineáris csökkenésre ültetett zajt tartalmaz. A lineáris komponens 15 értékről indul és 5 a végső értéke. A véletlenszerű komponens értéke ± 5.

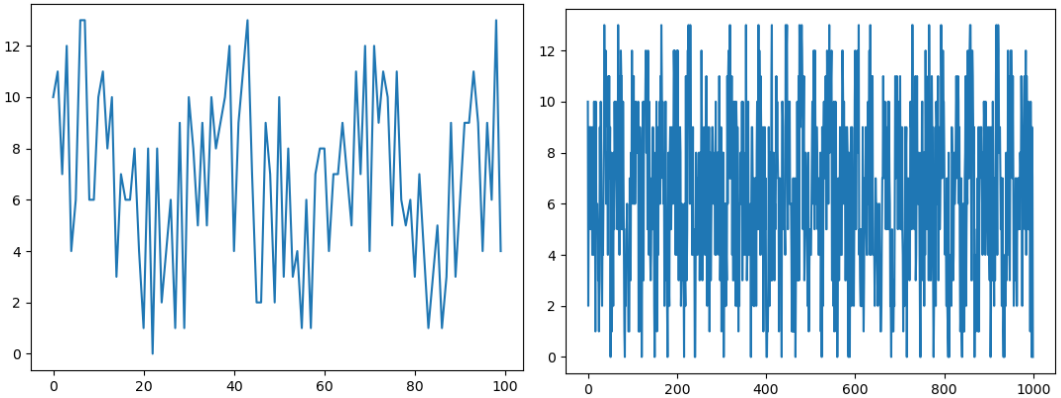
   Itt már az eltérés nagyobb, mint az előző esetben, ennek az oka az egészértékűség megszorítása és a minták számának növekedése.



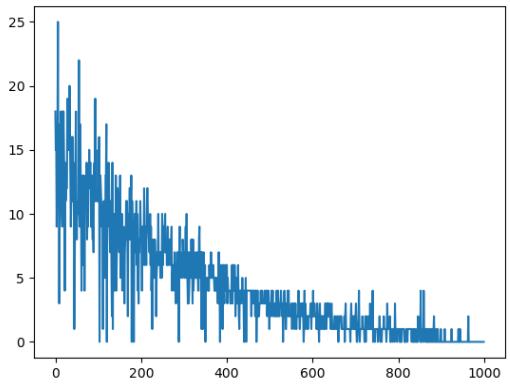
2. ábra: Véletlenszerű idősorok, $H_{100} = 0.5074, H_{1000} = 0.4933$



3. ábra: Idősorok lineáris trenddel, $H_{100} = 0.8451, H_{1000} = 0.9148$



4. ábra: Idősorok lineáris kis amplitúdójú szinuszos trenddel, $H_{100} = 0.5710, H_{1000} = 0.5637$



5. ábra: Valós céges idősor $H_{1000} = 0.9715$

3. Az idősor 0,...,10 véletlen értékeihez adjunk hozzá egy amplitúdóban eltolt $2\ \sin(x)$ függvényt!

   Látható, hogy a Hurst-exponens „észreveszi" az idősorban elrejtett szabályosságot. Ebben az esetben figyelnem kellett arra, hogy a 100 elemű és az 1000 elemű idősor szinuszos komponense azonos periódusú bemenettel legyen figyelembe véve, máskülönben nem ugyanazt a problémát vizsgáljuk.

A fentiek alapján látható, hogy a Hurst-analízis alkalmas az idősorok trendtartásának vizsgálatára.

Valós projekt vizsgálata

A következő idősor egy létező cég projektjéből származik. Az idősor alapját úgy határoztam meg, hogy az intervallumban a maximális hibaszám 25 legyen. Ezután ezzel az intervallum mérettel vizsgáltam a hiba előfordulást.

A trend rendkívül erős. Látható, hogy a projekt elején még sok a hiba, viszont az idő előre haladtával a hibák száma drasztikusan csökken.

Megjegyzés: a jelentős javulás oka részben az is, hogy a cég a kezdeti mérési eredményeket felhasználta a szoftverfejlesztés minőségének javítására.

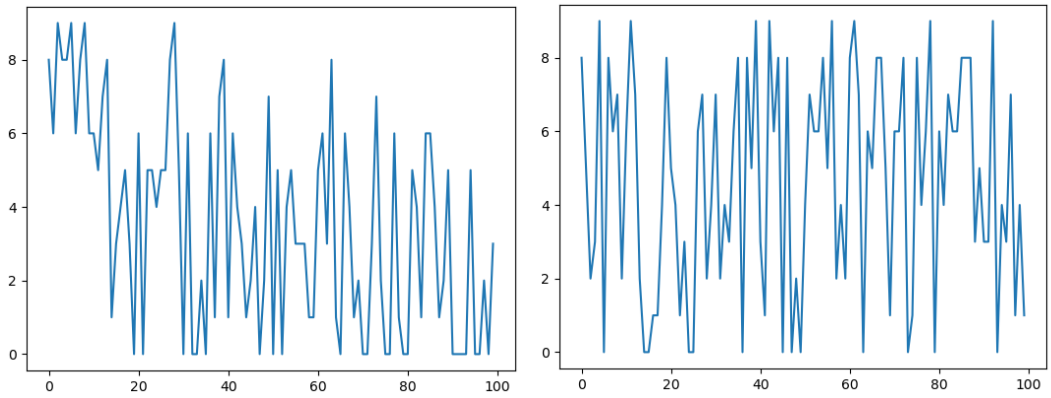
Bár számomra nem ez volt a cél, de hasznos eredménynek tekintem.

Hallgatói gyakorlatok trendvizsgálata

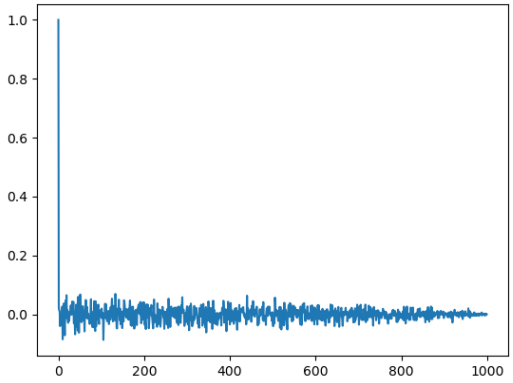
A hallgatói idősorok a következő eredményeket hozták. A 2. fejezetben említettem, hogy a kutatási módszertan eltér a javasolt céges vizsgálattól (lásd 7 oldal), de az eredmények jól tükrözik a vizsgált személyek teljesítményét az adott témakörben.

A jól teljesítő hallgató erős trendet mutat a hibák számának csökkenésében, ami komoly felkészülési munkát feltételez. Az összes hibaszám is ezt mutatja, ez a szám 355.

A hallgató laboratóriumi osztályzatai: 4 (jó) és 5 (jeles).



6. ábra: A jól teljesítő hallgató és a gyengébben teljesítő hallgató idősorai
$H_{good} = 0.7191, H_{bad} = 0.4688$



7. ábra: Véletlenszerű idősor autokorrelációs függvénye

A gyengébben teljesítő hallgató esetén ez nem tapasztalható, a Hurst-exponens nagy véletlenszerűséget mutat. Az összes hibaszám: 463, ami közel 30%-al több, mint a másik hallgatónál.

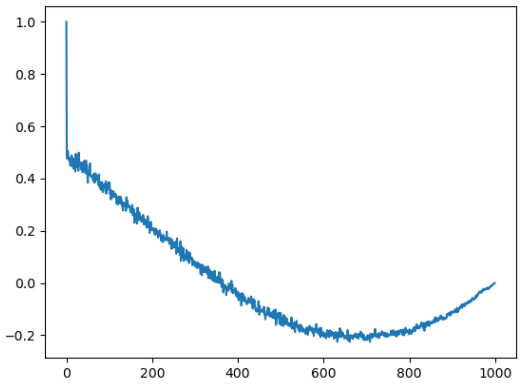
A hallgató laboratóriumi osztályzatai: 2 (elégséges) és 2 (elégséges).

3.3. Autokorrelációs vizsgálat

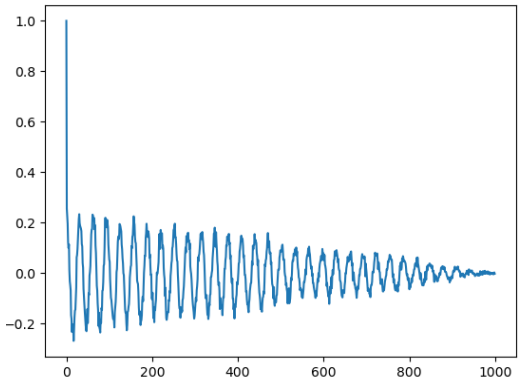
A 3. hipotézis szerint feltételezem, hogy ki tudom mutatni a gyártó minőségjavításra vonatkozó erőfeszítéseit az idősor autokorrelációs függvényének alakulásával[22].

A vizsgálathoz a szimulált idősorokat a valós cég által szolgáltatott idősort és a hallgatói idősorokat használom fel. A szimulált idősoroknál csak az 1000 elemű mintát tartalmazókat vizsgálom.

A 7. ábra a véletlenszerű idősor autokorrelációs függvényét mutatja. A függvény a 0. pontban egy értékű, ezután a függvény értéke a nulla körül mozog. Vagyis az egymást követő



8. ábra: Lineárisan csökkenő zajjal kevert idősor autokorrelációs függvénye



9. ábra: A szinuszos jellegű idősor zajjal kevert autokorrelációs függvénye

értékek egymástól függetlenek.

Matematikailag a 0. pozíciótól különböző helyeken a függvénynek 0 értékűnek kellene lennie, azonban az idősor véges és a Python programozási nyelv véletlen szám generátora sem valós véletlen szám generátor.

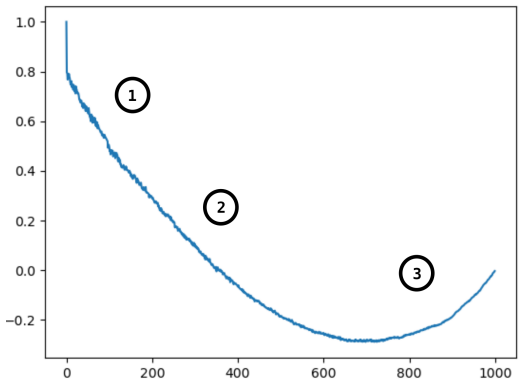
A 8. ábrán látható autokorrelációs függvényen látszik, hogy az egymást követő értékek függnek egymástól, mert a függvényérték az értelmezési tartomány túlnyomó részén különbözik a 0 értéktől. Ez megerősíti a trendtartást tulajdonságot is.

A 9. ábrán látható, hogy az autokorrelációs függvény periodikus jellegű. Ez azt jelenti, hogy az idősor többször is jelleget vált a szinuszos idősornak köszönhetően.

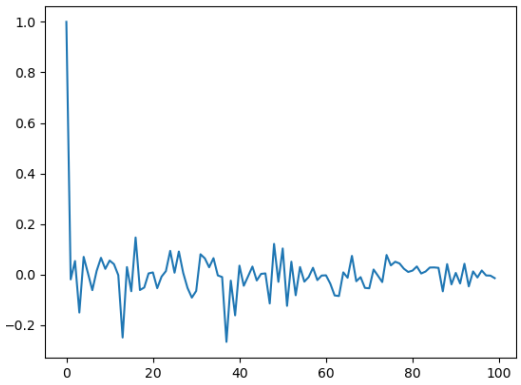
Kimutatható a trendtartás, de minőségi szempontból egy ilyen jellegű idősor nem megfelelő.

A 10. ábra a valós cég projektjének idősorának autokorrelációs függvényét mutatja. Az ábrán 3 jól elkülöníthető szakaszokat látunk, ezek:

1. egy meredek csökkenést, amely azt mutatja, hogy vannak független hibák,



10. ábra: Valós cég adott projektjének autokorrelációs függvénye



11. ábra: A gyengébben teljesítő hallgató idősorának autokorrelációs függvénye

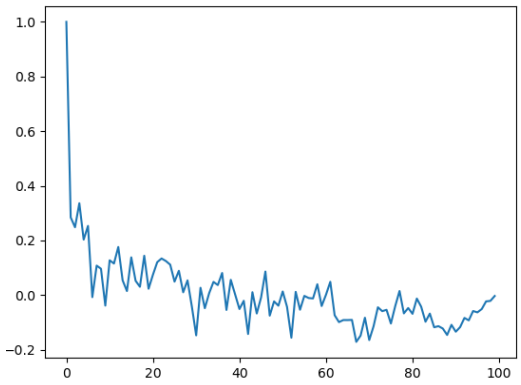
2. majd egy natív csökkenő szakaszt, amely a hibajelenségek jellegének megváltozását mutatja[23],
3. a harmadik szakasz a nulla értékre történő visszatérést mutatja, ami azzal magyarázható, hogy a tesztelés már a projekt zárásánál nem talál hibát.

Az 1. és a 2. szakasz találkozásnál egy viszonylag éles „töréspont” látható. Ez utasításban kifejezve azt jelenti, hogy ekkor végezték el az első hibaanalízist, majd ezt az intervallumot tartva rendszeresen felülvizsgálták a projektet.

A hallgatói idősorok elemzése is hasonló eredményt hoz.

A 11. ábrán a gyengébben teljesítő hallgató idősorának autokorrelációs függvényét láthatjuk. A kapott görbe jellegét tekintve a véletlenszerű idősor jellegét ismétli. A hallgató teljesítménye ez alapján véletlenszerűnek tekinthető, tehát ugyan a hallgató a minimumot teljesítette, de jelentős javulást nem mutatott.

A 12. ábrán a jól teljesítő hallgató idősorának autokorrelációs függvényét láthatjuk. A kapott görbe kimutatja, hogy a hallgató teljesítményében határozott összefüggés látható. Az



12. ábra: A jól teljesítő hallgató idősorának autokorrelációs függvénye

adatokból és az autokorrelációs függvényből kitűnik, hogy a hallgató erőfeszítéseket tett az eredményeinek javítására.

3.4. Regressziós vizsgálat

A kapott vizsgálati eredmények grafikus képe igen nehezen értelmezhető, nehéz különbséget tenni az egyes görbék között még akkor is, amikor már rendelkezésünkre áll a trendvizsgálat és az autokorrelációs vizsgálat eredménye. Ezért célszerű valamilyen „jelleggörbét” előállítani az idősorból. Erre használhatunk valamilyen regressziós függvényt.

Ebben a fejezetben megvizsgálom, hogy milyen jellegű regressziós közelítést érdemes alkalmazni.

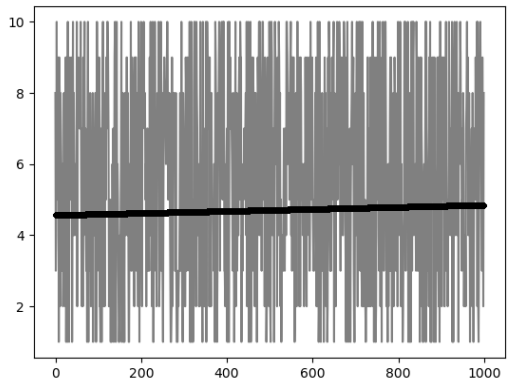
A vizsgált regressziós függvények:

1. exponenciális regresszió,
2. logisztikai görbe,
3. polinomiális regresszió.

Exponenciális regresszió

Az exponenciális regresszió alapvetően az ilyen jellegű idősorok vizsgálatára nem minden esetben alkalmas. Ennek két oka van:

1. az exponenciális függvény jellegét tekintve vagy szigorúan monoton növekvő, vagy szigorúan monoton csökkenő jellegű függvény, ha a teljesítmény hullámzik azt nem tudja megmutatni, azonban a jelleget igen,



13. ábra: Véletlen adatokból álló idősor exponenciális közelítése, $a = 4.567, b = 6.081\ 10^{-5}$

2. ha az idősor valamelyik eleme '0', vagy negatív értékű, akkor a regressziót kereső algoritmus hibát generál.

   Ezt a problémát úgy lehet megoldani, hogy az egész idősort egy konstans értékkel eltoljuk, hogy se '0' érték, se előjel váltás ne forduljon elő[7]. Én a probléma megoldására minden egyes elemhez '1' értéket adtam, majd a kapott exponenciális görbét ezzel az értékkel visszatoltam.

   $$v_i^{:} = v_i + 1, \quad \Rightarrow \quad y_i = a\, e^{b\,x} - 1, \text{ ahol: } a, b \text{ a regressziós paraméterek.}$$

   Ezzel a matematikai probléma megoldódott[24].

Így valójában nem tiszta exponenciális regressziót kaptam, de nem is ez a cél, a trend jellegét kell látni.

Nézzük az előzőekben bemutatott szimulált idősorok exponenciális görbékkel történő közelítését!

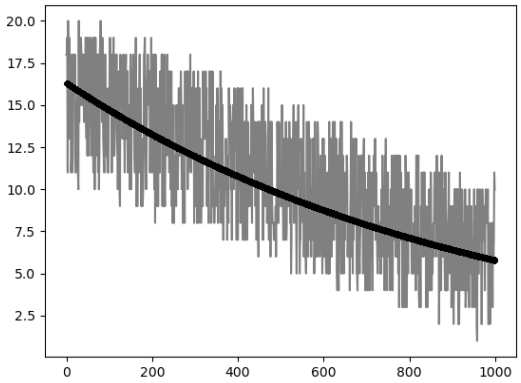
Az előzőekben említettem, hogy a regressziós függvény képlete $y = a\, e^{b\,x} - 1$. Ez alapján vizsgálom meg ennek viselkedését a szimulált adatsorokra.

A 13. ábrán látható véletlen idősort a regressziós függvény csak úgy tudja követni, hogy az adatsor teljes hosszában egy rendkívül lapos exponenciális függvényt állít elő. A b paraméter értéke $b = 6.081\ 10^{-5}$. Ezért az így kapott regressziós görbe gyakorlatilag egy egyenes.

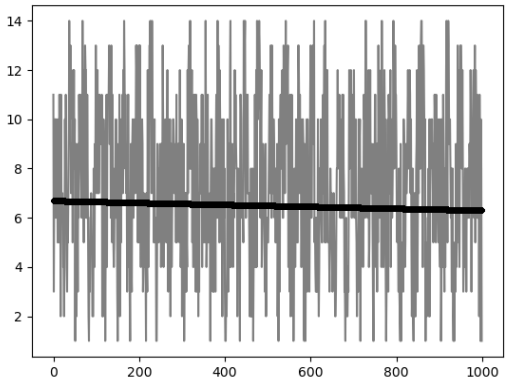
A 14. ábrán látható idősor határozott csökkenő trendet mutat, ezt a regressziós függvény úgy tudja követni, hogy a b értéke negatív lesz. A csökkenést pedig úgy, hogy a kitevő abszolút értéke jóval nagyobb, mint az előző esetben, $b = -1.036\ 10^{-3}$.

A szimulációban egyenletes eloszlású „zajt" adtam egy lineárisan csökkenő idősorhoz, az exponenciális függvény ezt nem tudja követni. Tehát ebben az esetben használata nem célszerű.

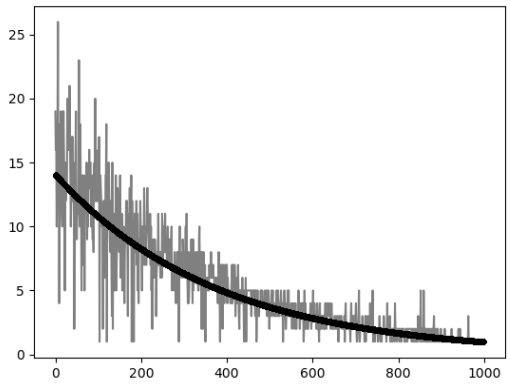
[7] Esetünkben csak a '0' érték jön szóba, mert negatív hibaszám nem létezik,



14. ábra: Lineáris trenddel rendelkező véletlen adatokkal kevert idősor exponenciális közelítése, $a = 16.31, b = -1.036\ 10^{-3}$



15. ábra: Szinusz jellegű, véletlen adatokkal kevert idősor exponenciális közelítése, $a = 6.69, b = -6.004\ 10^{-5}$



16. ábra: Valós céges adatok exponenciális közelítése,
$a = 14.062, b = -2.66\ 10^{-3}$

A 15. ábrán látható idősor szinuszos jelhez adott véletlen „zajt” tartalmaz. Ebben az esetben az exponenciális regresszió képtelen követni az idősort, így szintén egy lapos - gyakorlatilag egyenesnek tekinthető görbével közelíti. A b paraméter értéke $b = -6.004\ 10^{-5}$

A szimulált adatok után vizsgáljuk meg a valós idősorokat.

A 16. ábrán látható idősort az exponenciális közelítés elfogadható módon közelíti. Ennek két oka van:

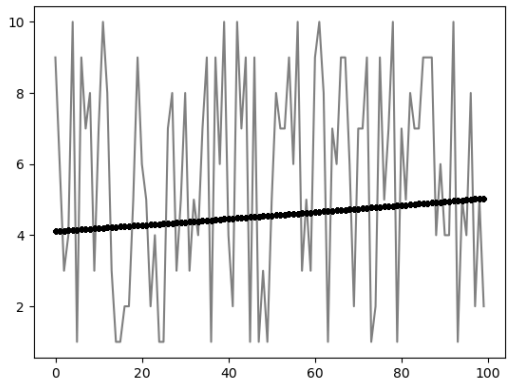
1. a vizsgált projektekben a dolgozó kollégák rendkívül tapasztalt programozók, így a folyamatos javulás elvárható és természetes,
2. a vizsgálati időszakban a részeredményeket felhasználták a folyamatok és a minőség javítására. Ezért látható az idősoron a jellegzetes lépcsőzöttség.

A hallgatói idősorok alátámasztják az előzőekben megállapított jellemzőket. Ebben az esetben az idősorok csak 100 eleműek.

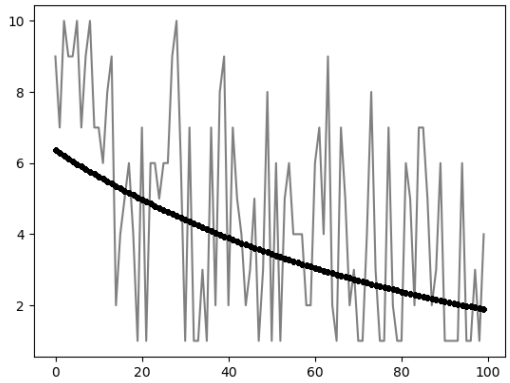
A 17. ábrán láthatjuk a gyengébben teljesítő hallgató idősorának exponenciális közelítését. Ez a görbe hasonló a 13. ábrán látható görbéhez .

Ez számunkra azt mondja, hogy a teljesítmény eléggé véletlenszerű. A b paraméter értéke $b = 2.052\ 10^{-3}$ az adatokhoz mérten kicsi.

A 18. ábrán láthatjuk a jobban teljesítő hallgató idősorának exponenciális közelítését. Ez a görbe hasonló a 16. ábrán látható görbével. Egyfajta javulást mutat, a b paraméter $b = -1.2210^{-2}$. A b negatív előjele mutatja a javulás tényét és abszolút értéke is közel hatszorosa az előző hallgató b értékének.



17. ábra: A gyengébben teljesítő hallgató idősorának exponenciális közelítése, $a = 4.11, b = 2.052\ 10^{-3}$



18. ábra: A jobban teljesítő hallgató idősorának exponenciális közelítése, $a = 6.355, b = 1.22\ 10^{-2}$

Összefoglalva:

- az exponenciális regresszió az erősen véletlenszerű idősorokat csak kis mértékben képes követni,
- az exponenciális regresszió a hullámzó jellegű adatokat nem képes követni,
- az exponenciális regresszió a csökkenő, vagy növekvő idősorokat képes jellemezni, de nem pontosan,
- az adott felhasználásban az exponenciális regresszió alkalmas lehet önjavító folyamatok idősorainak jellemzésére.

Következtetés:

Az exponenciális regresszió alkalmazása csak bizonyos esetekben indokolt.

Logisztikai görbe

A logisztikai görbe egy matematikai modell, amelyet növekedési folyamatok és populációk elemzésére használnak[25]. A görbét leíró úgynevezett logisztikai függvény, amely a következő:

$$y = \frac{c}{1 + a\, e^{-b\, x}}. \tag{1}$$

Létezik egy a sigmoid függvényre jobban támaszkodó logisztikai függvény is, amelynek a képlete[26]:

$$y = \frac{c}{1 + e^{-b\,(x-a)}}. \tag{2}$$

A két görbe között van látható eltérés, ezt a példákon be fogom mutatni.

Az oka, hogy nem a sigmoid változatot használom az, hogy a nagyon „zajos” idősorokat a sigmoid változat nem minden esetben kezelte jól.

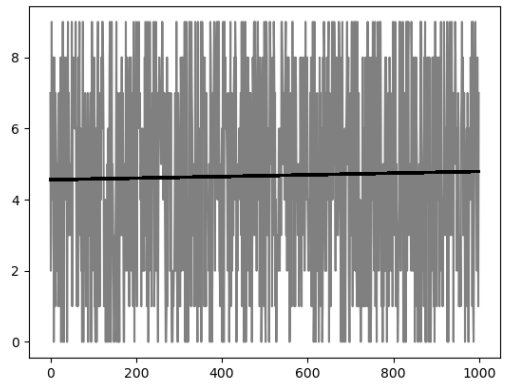
Mivel a logisztikai függvény egy konkáv - konvex, szigorúan monoton csökkenő, vagy konvex - konkáv, szigorúan monoton növekvő függvény az oszcilláló idősort (lásd 4. ábra) nem érdemes vizsgálni vele, mert ezt a a logisztikai görbe nem képes követni.

Vizsgáljuk a teljesen véletlenszerű idősort!

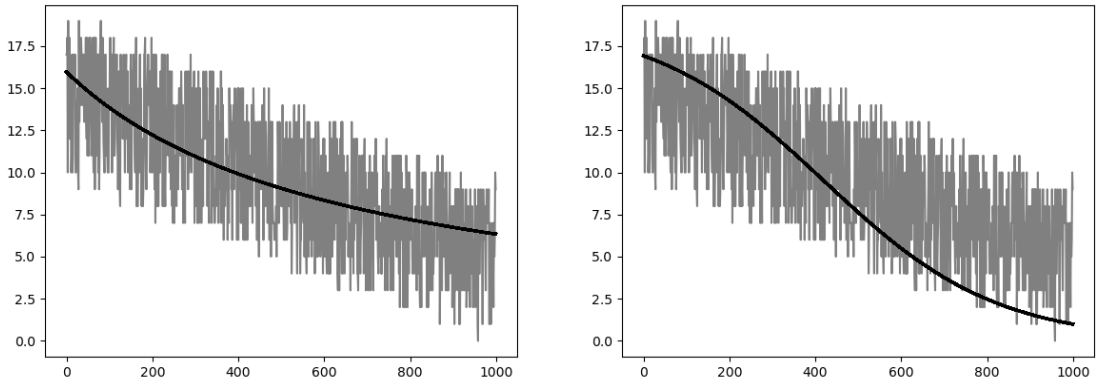
A 19. ábrán látható, hogy a logisztikai függvény az idősort nem tudja érdemben „értékelni”, ami természetesen helyes, az eljárások nagy része hibára is futott.

A görbe paraméterei: $a = -0.64, b = -2.66\,10^{-5}, c = 1.62$

A „zajjal kevert” lineáris trendet mindkét sigmoid függvény elfogadhatóan közelítette. Az eredmény a 20. ábrán látható.



19. ábra: Véletlenszerű idősor logisztikai függvénnyel közelítve



20. ábra: Logisztikai függvény és logisztikai sigmoid függvény

A baloldali görbe paraméterei: $a = -0.99, b = 1.51\ 10^{-5}, c = 0.15$ (1).
A jobboldali görbe paraméterei: $a = 5\ 10^{-3}, b = 417, c = 19.0$ (2).

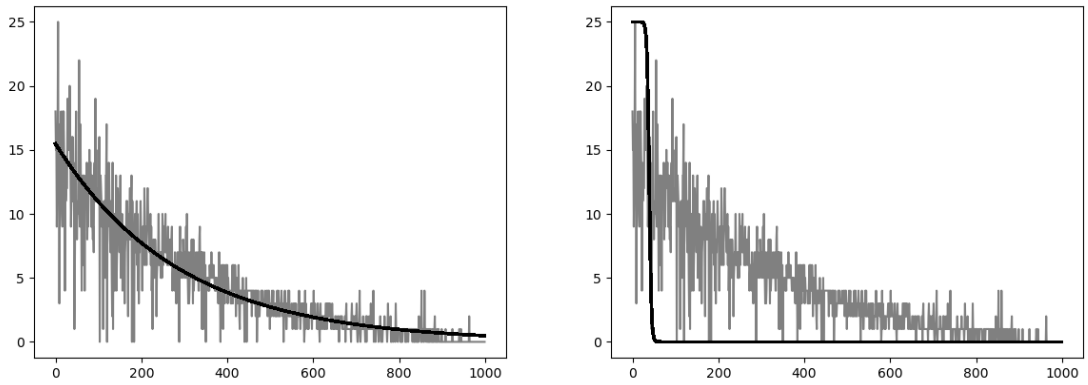
A 20. látható, hogy mindkét logisztikai függvény elég jól követi az idősort.

A valós céges adatokból álló idősort mindkét logisztikai függvénnyel vizsgáltam, lásd 21. ábra.

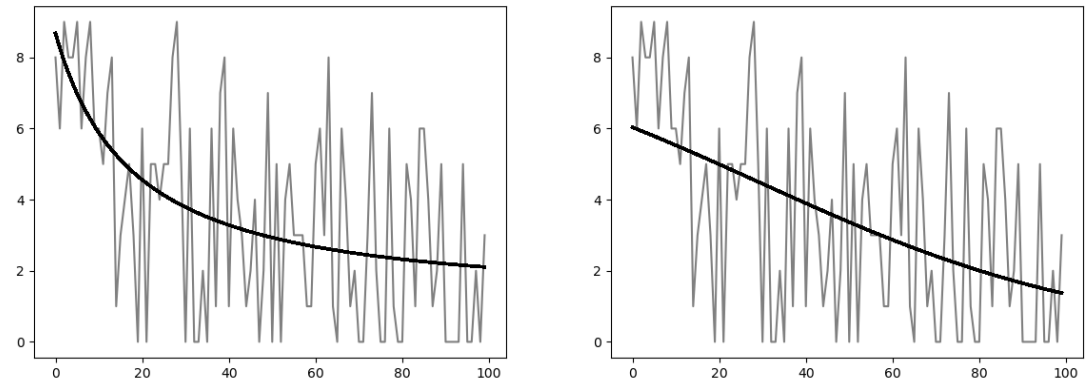
A baloldali görbe paraméterei: $a = -11108.46, b = -3.34\ 10^{-3}, c = 171917.39$ (1).
A jobboldali görbe paraméterei: $a = 0.40, b = 39, c = 25$ (2).

Az ábrán azt látjuk, hogy bár az idősor határozott trendet mutat, a sigmoid függvény használó logisztikai függvény nem követi az idősor jellegét.

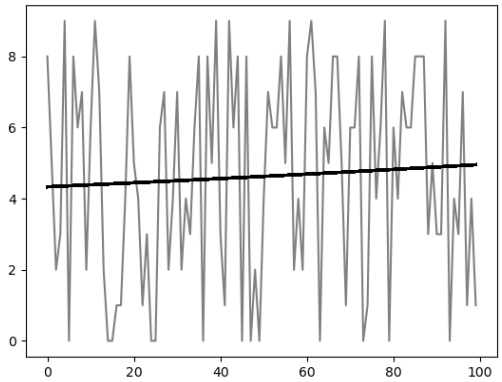
A gyengébben teljesítő hallgató idősora erősen véletlenszerű így a logisztikai sigmoid függvény nem kezeli jól. A másik 1 kifejezés által leírt logisztikai függvény általi közelítés a 22. ábrán látható.



21. ábra: Logisztikai függvény és logisztikai sigmoid függvény valós céges idősorra



23. ábra: A jól teljesítő hallgató idősorának közelítései



22. ábra: A gyengén teljesítő hallgató idősorának közelítése

A görbe paraméterei: $a = -0.969, b = -3.93\ 10^{-5}, c = 0.13$.

A jól teljesítő hallgató esetén mindkét típusú logisztikai függvény értékelhető eredményt ad, lásd 23. ábra.

A baloldali görbe paraméterei: $a = -0.831, b = 0.0101, c = 1.462$ (1).

A jobboldali görbe paraméterei: $a = 0.0245, b = 29, c = 9.0$ (2).

Összefoglalva:

- az erősen véletlenszerű idősorokat csak a (1) képlet alapján megadott logisztikai görbe tudja értelmezhető módon követni,
- a logisztikai görbe két vizsgált eljárása a hullámzó jellegű idősorokat a szigorúan monoton jelleg miatt nem tudja követni,
- a logisztikai görbe két vizsgált eljárása erős trend jellegű folyamatokat képes jól követni, de nem minden esetben. A sigmoid alapú idősor nagy eltérést mutathat.
- a (1) képletet megvalósító idősor görbéje nagyon hasonló az exponenciális regresszió görbéjéhez, de nagyobb számítási erőfeszítést igényel,
- a logisztikai görbe alkalmas lehet önjavító folyamatok jellemzésére.

Következtetés:

a logisztikai görbe két vizsgált eljárása közül a (1) képlet alapján számított eljárás több helyen alkalmazható. A logisztikai görbe általam vizsgált két változata minden esetben egyéni vizsgálatot igényel.

Polinomiális regresszió

Az idősorokat közelíthetjük polinomokkal. Ebben az esetben az a kérdés, hogy mekkora legyen a polinom fokszáma.

Az 1-nél magasabb fokszámú polinomok lehetővé teszik az idősor hullámzásának korlátozott követését. Ez fontos lehet a döntéshozó számára, ha ez a hullámzási tulajdonság túl magas, akkor várható, hogy a jövőben a gyártó továbbra is ilyen hullámzó teljesítményt mutat, ami problémát jelent.

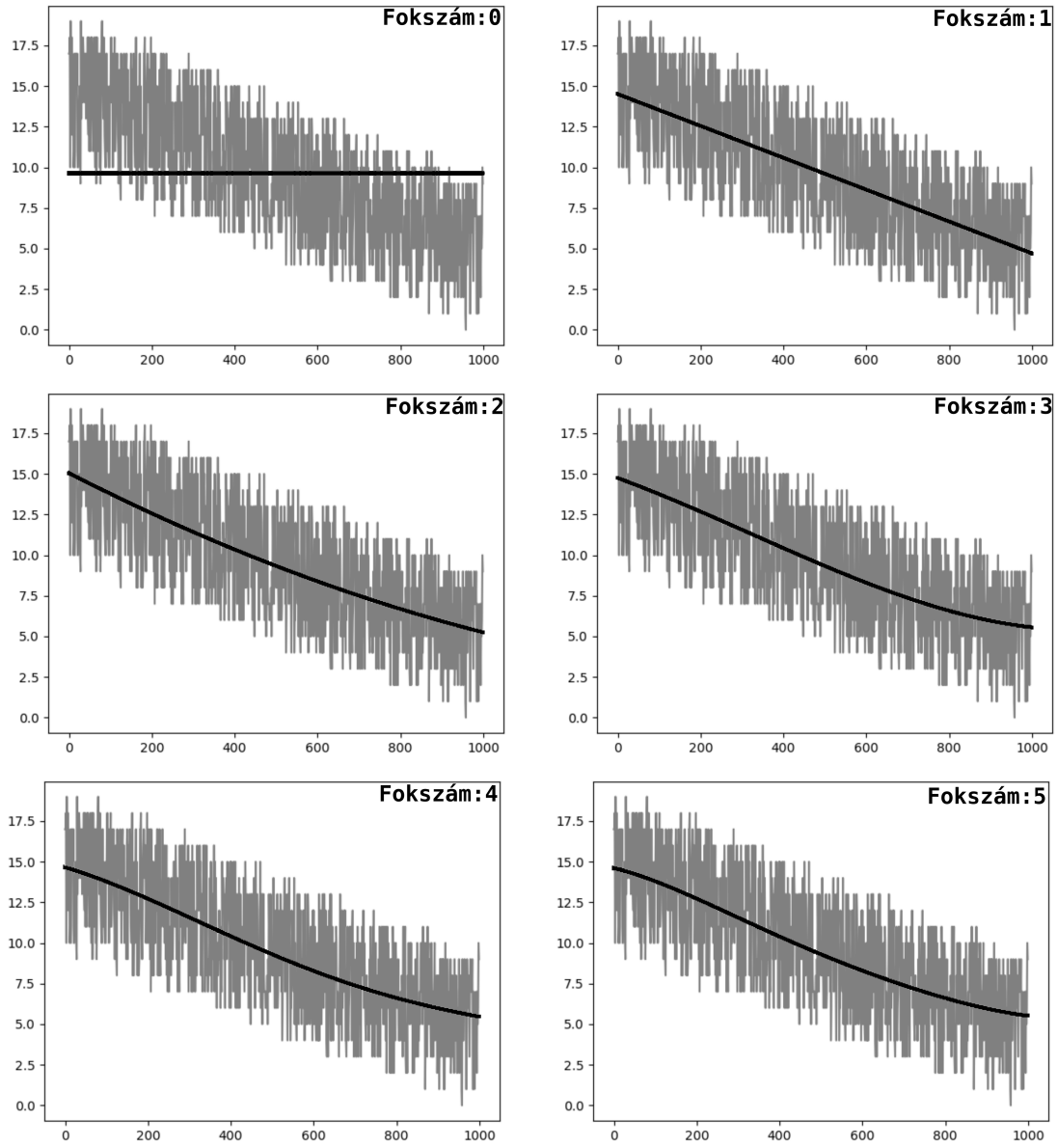
A vizsgálatból itt szintén kizártam a 4. ábrán látható idősort, amely szinuszos jellemzőkkel rendelkezik.

A vizsgálatot elvégeztem a 3. ábrán látható lineáris trenddel rendelkező „zajjal kevert” idősorra. Az idősort 0. fokú közelítéstől 5. fokú közelítésig a 24. ábra mutatja.

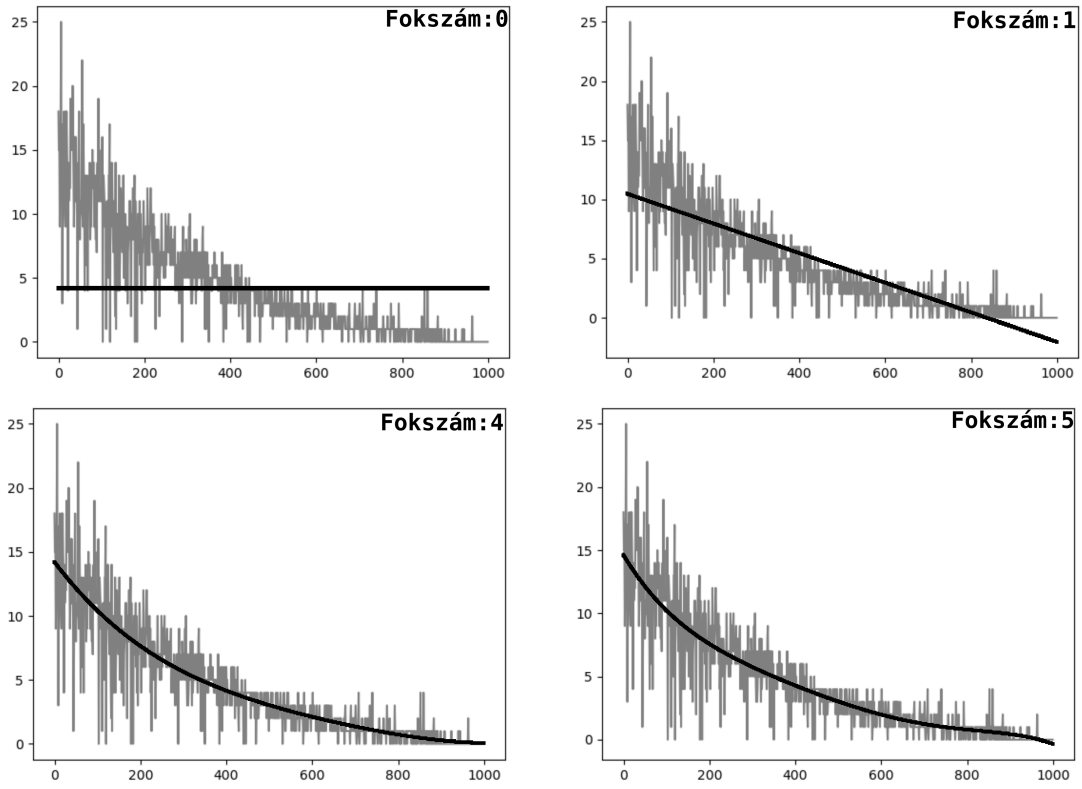
A 24. ábrán látható diagramok esetén az idősor alapját ismerjük. Ez egy egyenes, amelyre egyenletes eloszlású véletlen „zajt” kevertem. Így könnyen vizsgálhatom a a regressziós polinomokat a fokszám függvényében. A következő megállapításokat teszem:

0. fokú közelítés esetén a kapott egyenes az idősor átlagát adja meg,

1. fokú közelítés a globális trendet reprezentálja, látható, hogy az idősor honnan indul és hová tart[27],



24. ábra: Lineáris trenddel rendelkező idősor 0.-5. fokú polinomiális közelítése



25. ábra: Valós céges idősor közelítése regressziós polinomokkal

2. és 3. fokú közelítés ebben az esetben a regressziós polinom már képes az esetleges hullámzásokat követni. Azonban a görbe lokális szélsőértékeinek maximális száma a fokszám mínusz egy értékű. Ez lehet jó, de az esetek többségében kevés,

4. 5. fokú közelítés ebben az esetben a regressziós polinom képes követni a hullámosságot[28], ha az nem túlságosan „gyakori”.

Ha az idősor nagyon „hullámzik”, akkor az az általam végzett vizsgálat szempontjából nem megfelelő. Ezért az elemzés ezen pontján a 4. és az 5. fokú regressziós polinom a célnak megfelelő.

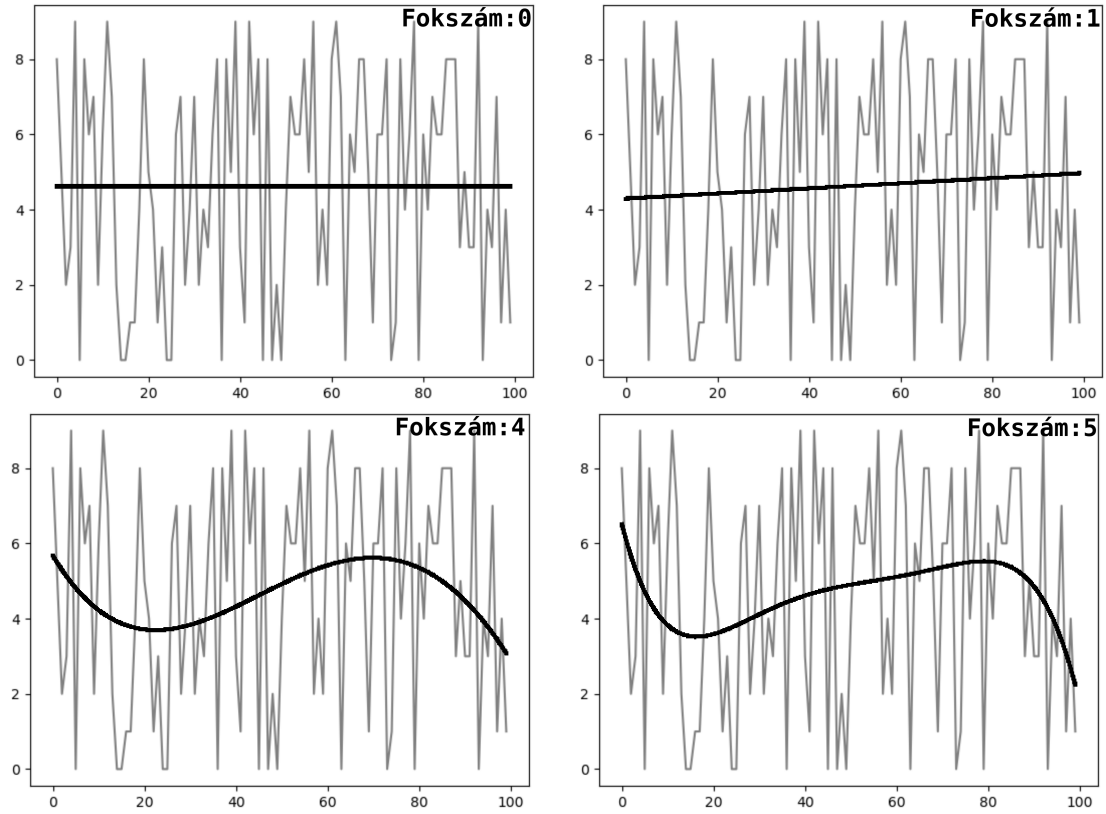
A további vizsgálatokból a 2. és 3. fokú közelítéseket kizártam.

A 25. ábrán valós céges idősor közelítése látható regressziós polinomokkal.

A 25. ábrán látható 0. fokszámú diagram megadja az idősor átlagát, tehát a kiválasztott vizsgálati intervallumban elkövetett hibák átlagos számát.

Ez a valós, céges idősorban 4.28-as érték. A vizsgálatban ez lehet egy elsődleges vizsgálati kritérium.

Az 1. fokszámú közelítés jellemzi az általános trendet, a 25. ábra alapján egyértelmű a hibák számának drasztikus csökkenése.



26. ábra: A gyengébben teljesítő hallgató idősorának közelítése polinomokkal

A 4. és az 5. közelítés jellemzi a vizsgálati folyamat alatt a trendeket. Ha vizsgáljuk a görbéket a hibaszám csökkenése emlékeztet az exponenciális görbére. Ennek oka az, hogy a cég a vizsgálatok eredményét már a fejlesztési folyamatban felhasználta a minőség javítására.

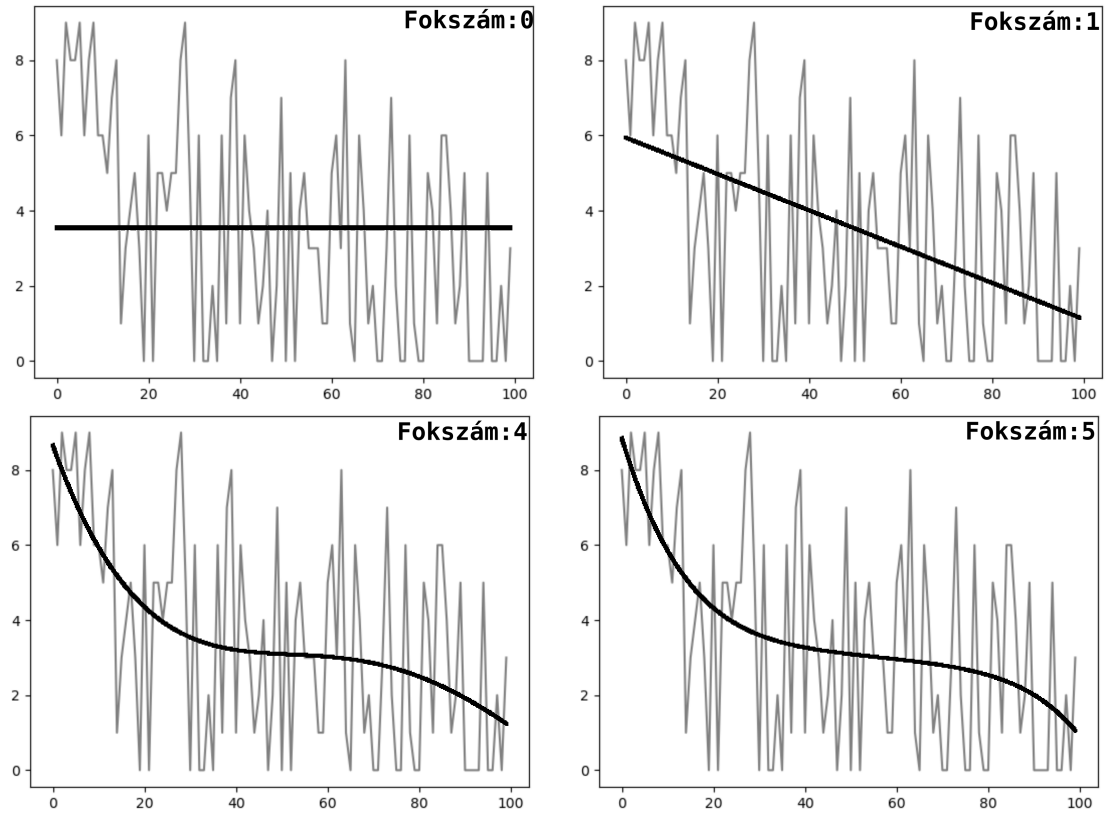
Láthatóan a 4. 5. fokszámú közelítés között minimális a látható különbség, egyedül a 5. fokú görbe mutat a végén további csökkenést. Ez azt jelenti, hogy az 5. fokú görbe képes kimutatni a végső eredményeket is.

A hallgatói idősorok polinomiális közelítései a 26. és a 27. ábrán láthatók.

A 26. ábrán látható hallgató teljesítményét érdekes módon jellemzik a különböző regressziós polinomok. A 0. fokú érték magasabb, mint a 27. ábrán látható hallgatónál. Az 1. fokú közelítés egyenesen romló trendet mutat.

Azonban a 4. és 5. fokú görbe egyértelműen mutatja az adott időszakokban a javulást. Az első lokális minimum az első félév végén található, illetve a görbék végén egy erőteljes csökkenés látható. Amit az is jellemez, hogy a hallgató mindkét félév végén elégséges osztályzatot kapott.

A 27. ábrán látható hallgató teljesítményét jól jellemezhetjük a görbék alapján. A hibaráta (0. fokú közelítés) alacsonyabb, mint az előző hallgatóé. Az általános trend (1. fokú közelítés)



27. ábra: A jól teljesítő hallgató idősorának közelítése polinomokkal

igen határozott. A 4. és 5. fokú görbe nem mutat lényeges különbséget.

Érdekes jelenség mindkét hallgató esetén a 4. és 5. fokú görbén középen ahol vagy növekszik a hibák száma, vagy majdnem változatlan. Itt a kérdéses regressziós görbék vagy közel vízszintesek, vagy emelkednek. Ennek az az oka, hogy az első és a második vizsgált félév között egy nyári időszak van, ami elég hosszú és ez a hallgatók esetén a „felejtés időszaka”.

Összefoglalva:

- a 0. fokú regressziós polinom tulajdonképpen a hibaátlagot adja meg, amely elsődleges mutatónak tekinthető,
- az 1. fokú regressziós polinom az idősor globális trendjét mutatja. A Hurst- exponens a trendtartás mértéket, mig ez a diagram a trend irányát adja meg,
- a 2. és 3. fokú regressziós polinomok használata nem célszerű,
- a 4. és 5. fokú regressziós polinomok az idősor belső tulajdonságát is jelzik. Ez a hallgatói idősorokban látszik jól,
- a magasabb fokú közelítéseket nem vizsgáltam, mert a a 4. és 5. fokú regressziós görbék kielégítő információt szolgáltattak.

Következtetés:

a vizsgált eljárások közül a regressziós polinomok alkalmazása bizonyult a legjobbnak. A szimulációk és a mérések során a 0. és az 1. fokú közelítés hasznos információt nyújt. A 4. és az 5. fokú regressziós polinomok képesek megmutatni az idősorok belső tulajdonságait.

4. Tézisek

Tézis 1. Megállapítottam és kísérleti mérésekkel igazoltam, hogy a programfutás időalapja a gépi utasítások súlyozott számával helyettesíthető és összehasonlítható eltérő architektúrák és fejlesztői rendszerek esetén.

Lásd 3.1. fejezet!

Tézis 2. Kimutattam, hogy két gyártó közötti minőségi különbség kimutatható a hiba-előfordulás idősor trendjének összehasonlításával.

Az összehasonlítást számszerűen a Hurst-exponens mutatja ki, a trend globális jellegét a lineáris regresszióval lehet meghatározni.

Lásd 3.2. és 3.4. fejezet!

Tézis 3. Kimutattam, hogy a gyártó minőség javulásra fordított erőfeszítései összehasonlítási szempontból vizsgálhatók az idősorok autokorrelációs függvényeinek összehasonlításával.

Lásd 3.3. fejezet!

Tézis 4. Kimutattam, hogy az idősorok jellegének alakulása jól közelíthető 4., illetve 5. fokú regressziós polinomokkal.

Lásd 3.4. fejezet!

Alkalmazási javaslatok

Esettanulmány szoftvermegbízhatóságra

Az eredeti ötlet az volt, hogy egy biztonságkritikus szoftverprojekt megvalósításához gyártót szeretnénk választani. Ez az esettanulmány egy hipotetikus szituációt mutat be, hogy a kiválasztási eljárást az előző fejezetekben ismertetett módszerekkel hogyan lehet megvalósítani.

A következőkben minden adat és diagram feltételezett.

A feltételezett helyzet az, hogy három lehetséges gyártónk van, legyen a nevük 'A', 'B' és 'C'. Mindhárom egyedi fejlesztéssel foglalkozik és mindhárom gyártó gyűjti a fejlesztési adataikat az 1. tézisnek megfelelő módon, azonos szempontok szerint.

Az átlagos hibaráta E_{rx}:

$E_{rA} = 2.1\ 10^{-25}$,

$E_{rB} = 2.4\ 10^{-25}$,

$E_{rC} = 1.9\ 10^{-23}$.

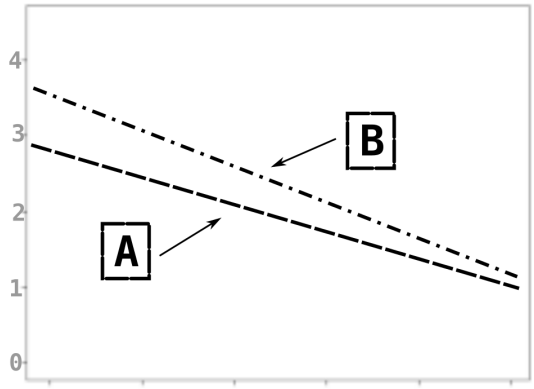
Mivel a 'C' gyártó hibarátája közel két nagyságrenddel magasabb, mint 'A' és 'B' gyártóé, így 'C' gyártó már kiesett a jelöltek közül.

Ezután a trendtartás vizsgálat következik a 2. tézisnek megfelelően. A Hurst-exponensek H_x:

$H_A = 0.973$,

$H_B = 0.976$.

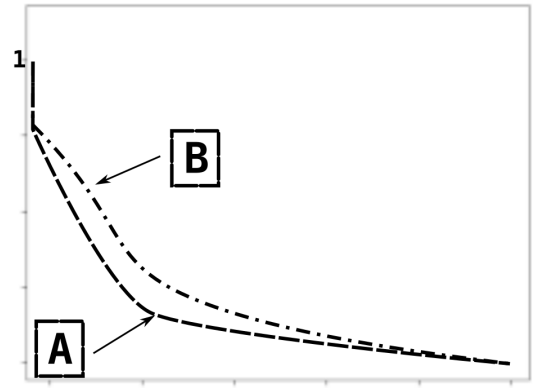
A különbség 'A' javára $3\ 10^{-3}$. Természetesen meg kell vizsgálnunk a trend jellegét, ezt a 28. ábra mutatja.



28. ábra: ’A’ és ’B’ gyártó regressziós egyenesei

Látható, hogy ugyan ’A’ értékei alacsonyabbak, de ’B’ diagramja meredekebben csökken.

Ekkor megvizsgálhatjuk az idősorok autokorrelációs függvényeit a 3. tézis szerint. Az autokorrelációs függvények a 29. ábrán láthatók[8].

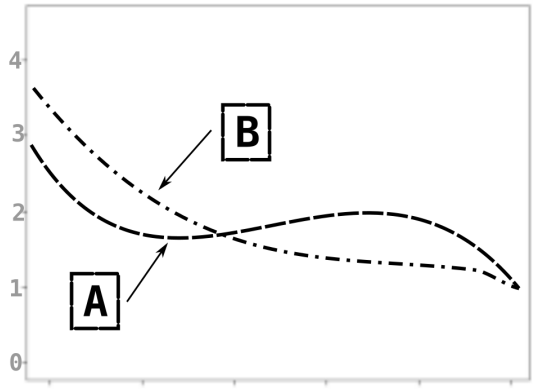


29. ábra: ’A’ és ’B’ autokorrelációs függvényei

’B’ gyártó görbéje egy kicsit magasabb, mint ’A’ gyártóé, ezért valószínűbb, hogy ’B’ gyártó több erőfeszítést tesz a minőség javítására.

Az eddigiek alapján a megrendelő dolga nem egyértelmű. Ezért a 4. tézisben tárgyalt regressziós polinomok vizsgálatát is célszerű elvégezni. A feltételezett 5. fokú regressziós görbét láthatjuk a 30. ábrán.

[8]Hangsúlyozom, hogy ezek ebben az esetben kitalált értékek, az autokorrelációs függvény soha nem lesz ilyen sima



30. ábra: ’A’ és ’B’ 5. fokú regressziós görbéi

A kérdéses esetben azt látjuk, hogy az ’A’ gyártó regressziós görbéje tartalmaz egy emelkedő szakaszt, míg a ’B’ gyártó regressziós görbéje szigorúan monoton csökken. A másik észrevehető tulajdonság, hogy az ’A’ és a ’B’ gyártók regressziós görbéi azonos értéken végződnek.

A megrendelőnek ez a következő információt tartalmazza:

- az ’A’ gyártó induló értéke alacsonyabb,
- a végső értékek ugyanazok,
- az ’A’ gyártó minősége hullámzó,
- a ’B’ gyártó folyamatos javulást mutat.

Ez azt jelenti, hogy a ’B’ gyártó várható szoftverminősége valószínűleg jobb lesz, mint ’A’ gyártó szoftverminősége.

Ezt alátámasztja a trendvizsgálat és az autokorrelációs vizsgálat. Némileg ellentmond az átlagos hibaráta.

Egy másodlagos eredmény

Mivel csak egyetlen cégtől kaptam értékelhető adatokat, ezért hallgatókat vizsgáltam. Ez sok szempontból segített a kutatásban, de egy másodlagos eredmény is született.

A vizsgálatok során az összehasonlíthatóság nagyon fontos szempont volt, ezért annyira erős megszorításokat alkalmaztam, hogy igen kevés hallgató felelt meg ezeknek.

Az eredmények jól tükrözték a hallgatók teljesítményét, bár a jobban teljesítő hallgató esetén várható lett volna a két jeles osztályzat, de az csak jó és jeles lett.

Sajnos a számonkérésünk vizsga és zárthelyi centrikus, ez viszont csak pillanatnyi teljesítményt mér. Mondhatjuk pillanatnyi formát mér.

Definíció 5. Formának nevezzük a tudás pillanatnyi állapotát, amelyet számos külső tényező is befolyásol.

Tehát a forma nem a valós tudás, hanem annak csak egyfajta vetülete. Erre persze mondhatjuk, hogy a százméteres síkfutást is a versenyen mérjék le, de az oktatás és a számonkérés nem sporttevékenység.

Programozóként, teszterként, projekt vezetőként ismerem az úgynevezett black-out jelenséget. Ez azt jelenti, hogy a programozó egy teljesen nyilvánvaló hibát képtelen felismerni, hiába van a szeme előtt, főleg ha tapasztalatlan. Ekkor megbuktatjuk a hallgatót? Igen van ilyen „oktató".

Viszont ha az ismertetett módon vizsgáljuk a hallgatók teljesítményét, akkor sokkal valóságosabb képet kaphatunk a valós tudásukról.

Felmerült egy másik mérési lehetőség. Ebben az esetben a hallgatókat nem egyenként, hanem csoportonként vizsgáljuk. Ekkor nem a hallgatókat, hanem az oktatót minősítjük.

Azonban ekkor nem elegendő két félévet vizsgálni, mert a hallgató anyag tudása erősen fluktuálódhat. Becslésem szerint 10 félévnyi adatgyűjtés szükséges. Ez nagyjából megfelel az oktatók minősítési periódusának.

Ezt a vizsgálatot azonban a pandémia meghiúsította.

Összefoglalás és a továbbfejlesztés lehetősége

A kutatás egy ötletből és Szilágyi Győző Attila "A légi balesetek fraktáldimenziója" című konferencia előadásának meghallgatásából[20] indult. Akkor fogalmazódott meg bennem, hogy olyan statisztikai vizsgálatokat végezzek el idősorokon, amelyek megkönnyítik a szoftverminőség előzetes becslését.

Első pillanatban éreztem, hogy a nagy szoftver cégek nem szívesen adják ki a kért adatokat, azonban nem számoltam azzal, hogy egyetlen kis cégtől kapok csak releváns információkat. Ez a cég a mérési eredményeket már a vizsgálat során felhasználta a minőség javítására. Ennek egyenes következménye az lett, hogy 4 éves időszakban összesen négy hibajelentés keletkezett. Ebből egy volt valós programhiba, a három további hibajelentés kezelési eltérés volt, ami nem érintette a működést.

A cégeknek érdemes lenne ezeket a vizsgálatokat bevezetni, mert segíthetne a minőség javításában. Az alapvetően érthető, hogy ezeket az adatokat nem szívesen adják ki, hiszen akár negatív hatással is lehet egy esetleges projekt elnyerésében.

A hallgatói vizsgálatok érdekes eredményeket hoztak. A kérdéses hallgatók eredményei jól tükrözik az idősorok által mutatott jellemzőket. Az egyértelmű, hogy itt is sokkal több vizsgálatra lenne szükség, de ez sok időbe kerül.

Továbbfejlesztés lehetőségei

Kezdjük a végével! A hallgatói idősorokat tanulói csoportokra alkalmazva a módszer az oktatók minősítésére is használható. Ezt a vizsgálatot, mint említettem a pandémia meghiúsította.

A továbbfejlesztés további lehetősége a mesterséges intelligencia mélytanuló eljárásainak alkalmazása. Ha van kellő tanítási minta, az idősor jellegéből egy ilyen eljárás képes felismerni egy mintát és időben figyelmeztetni a beavatkozókat a jelentkező problémákra.

A vizsgálatok arra irányúltak, hogy a gyártókat össze tudjuk hasonlítani. A kutatás új iránya lehet, hogy ne csak összehasonlításra tudjuk a módszert alkalmazni, hanem számszerű becslést is tudjunk előállítani. Ez kockázat elemzési szempontból ugrásszerű fejlődés lenne.

Ábrajegyzék

Irodalomjegyzék

Saját hivatkozások

[1] r. Schuster György (ÓE-KVK): ” Szoftver minőség előre becslés idősorok alapján”, Forrás: KVK Habilitációs Workshop Minikonferencia Kiadvány kötet, ISBN 978-963-449-317-4, Elérhető: https://habilworkshop2023.kvk.uni-obuda.hu/static/2023/05/31/KVK_habil_workshop_kiadvanykotet_2023_lezart_ok.pdf 75-82

[2] Schuster György, Ady László: ”BIZTONSÁGKRITIKUS SZOFTVER FEJLESZTÉS”, Forrás: Repüléstudományi közlemények30. évf. 1. sz. (2018.), Elérhető: https://epa.oszk.hu/02600/02694/00076/pdf/ EPA02694_rtk_2018_01_151-160.pdf

[3] Schuster György: Szoftvertechnológia”, Elérhető: https://uni-obuda.hu/users/schuster.gyorgy/szoftvertech_kh.pdf 6. oldal

[4] Schuster György: ”Szoftvertechnológia”, Elérhető: https://uni-obuda.hu/users/schuster.gyorgy/szoftvertech_kh.pdf 58. oldal

[5] György Schuster, Daniel Tokody, Imre János Mezei: ”Software Reliability of Complex Systems Focus for Intelligent Vehicles”, Forrás: Conference paper 25 March 2017, Vehicle and Automotive Engineering pp 309–321, Elérhető: https://link.springer.com/chapter/10.1007/978-3-319-51189-4_28

Felhasznált irodalom

[6] NTSB: ”Aviation Investigation Final Report”, Forrás: NTSB DCA19LA154, Elérhető: https://data.ntsb.gov/carol-repgen/api/Aviation/ReportMain/GenerateNewestReport/99537/pdf

[7] IT governance: ”Software Capability Maturity Model (CMM)”, Forrás: IT Governance UK, Elérhető: https://www.itgovernance.co.uk/capability-maturity-model

[8] IT governance: ”Software Capability Maturity Model (CMM)”, Forrás: IT Governance UK, Elérhető: https://www.itgovernance.co.uk/capability-maturity-model

[9] Stephen Watts: ”CMMI: An Introduction to Capability Maturity Model Integration” (19 February 2020), Forrás: BMC Elérhető: https://www.bmc.com/blogs/cmmi-capability-maturity-model-integration/

[10] M.E., Fagan (1976): ”Design and Code inspections to reduce errors in program development” Forrás: IBM Systems Journal. 15 (3): 182–211., Elérhető: http://www.mfagan.com/pdfs/ibmfagan.pdf

[11] Lori Kinney (2023): How Fagan Style Software Inspection Can Benefit Your Business, Forrás: isixsigma, Elérhető: https://www.isixsigma.com/dictionary/fagan-style-software-inspection/

[12] Christopher Grafton: ”Mastering Hurst Cycle Analysis: A modern treatment of Hurst’s original system of financial market analysis”, Forrás: Google könyvek (2011), Elérhető: https://books.google.hu/books?id=q-1jAgAAQBAJ&hl=hu&source=gbs_navlinks_s

[13] George E. P. Box, Gwilym M. Jenkins: ”Time Series Analysis: Forecasting and Control”, Forrás: Google könyvek (1976), Elérhető: https://books.google.hu/books/about/Time_Series_Analysis.html?id=1WVHAAAAMAAJ&redir_esc=y

[14] Alan R. Weiss: ”Dhrystone Benchmark History, Analysis, ”Scores” and Recommendations White Paper October 1, 2002” Forrás: ECL, LLC 6507 Jester Blvd 2222 Francisco Drive Suite 511 Suite 510-203 Austin, Texas 78750 El Dorado Hills, California 95762 Elérhető: https://www.eembc.org/techlit/datasheets/dhrystone_wp.pdf

[15] RFC 3174 - US Secure Hash Algorithm 1 (SHA1) Forrás: faqs.org Elérhető: http://www.faqs.org/rfcs/rfc3174.html

[16] U. Blumenthal, F. Maino: ”The Advanced Encryption Standard (AES) Cipher Algorithm in the SNMP User-based Security Model” Forrás: Network Working Group Request for Comments: 3826 Lucent Technologies Category: Standards Track Elérhető: https://datatracker.ietf.org/doc/html/rfc3826

[17] K. Moriarty, B. Kaliski, J. Jonsson, A. Rusch: ”PKCS #1: RSA Cryptography Specifications Version 2.2, November 2016”, Forrás: Internet Engineering Task Force (IETF) Request for Comments: 8017 Elérhető: https://datatracker.ietf.org/doc/html/rfc8017

[18] Friedl Katalin: ”Nagyságrendek Kiegészítő anyag az Algoritmuselmélet tárgyhoz”, Forrás: BME SzIT 2022. február 19., Elérhető: www.cs.bme.hu/ friedl/alg/nagysagrend.pdf

[19] Subir Mansukhani: ”The Hurst Exponent: Predictability of Time Series 2012”, Forrás: Analytics Elérhető: https://pubsonline.informs.org/do/10.1287/LYTX.2012.04.05/full/

[20] Szilágyi Győző Attila: ”A LÉGI BALESETEK FRAKTÁLDIMENZIÓJA”, Forrás: REPÜLÉSTUDOMÁNYI KÖZLEMÉNYEK XXVIII. ÉVFOLYAM 2016, Elérhető: https://www.repulestudomany.hu/folyoirat/2016_2/2016-2-14-0311_Szilagyi_Gyozo_Attila.pdf

[21] Vibhu Singh, Varun Divakar and Ashish Garg: ”Hurst Exponent: Calculation, Values and More” Oct 29, 2018 , Forrás: Quantinsti, Elérhető: https://blog.quantinsti.com/hurst-exponent/

[22] Rob J. Hyndman and George Athanasopoulos: (2016) ”Forecasting: Principles and Practice, chapter 2.3”, Forrás: Monash University, Australia Elérhető: https://otexts.com/fpp2/autocorrelation.html

[23] PennState Eberly College of Science Deprtment of Statiastics: ”Stat 501 Regression Methods, Topic 2: Time Series & Autocorrelation” Forrás: PennState Eberly College of Science Elérhető: https://online.stat.psu.edu/stat501/lesson/topic-2-time-series-autocorrelation

[24] PennState Eberly College of Science: ”12.6 - Exponential Regression Example” 2023, Forrás: PennState Eberly College of Science Applied Regression Analysis STAT462, Elérhető: https://online.stat.psu.edu/stat462/node/205/

[25] David W. Hosmer Jr., Stanley Lemeshow, Rodney X. Sturdivant: ”Applied Logistic Regression”, Interpretation of the Fitted Logistic Regression Model (Pages: 49-88), ISBN:9781118548387, Forrás: Book Series:Wiley Series in Probability and Statistics, Elérhető: https://onlinelibrary.wiley.com/doi/10.1002/9781118548387.ch3,

[26] Samprit Chatterjee, Jeffrey S. Simonoff: ”Handbook of Regression Analysis”, Forrás: 2013 by John Wiley & Sons, ISBN: 978-0-470-88716-5 Elérhető: https://ocd.lcwu.edu.pk/cfiles/Statistics/Stat-503/HandbookofRegressionAnalysisbySampritChatterjeeJeffreyS.Simonoffauth.z-lib.org.pdf

[27] Douglas C. Montgomery, Elizabeth A. Peck, G. Geoffrey Vining: ”Introduction to Linear Regression Analysis” (2012), ISBN-0470542810, Forrás: WJILEY, Elérhető: https://www.amazon.com/Introduction-Regression-Analysis-Douglas-Montgomery/dp/0470542810 (vásárolható)

[28] J. Fan, I Gijbels: ”Local Polynomial Modelling and Its Applications: Monographs on Statistics ...”, (1996), ISBN 9780412983214, Forrás: CRC press, Elérhető: https://books.google.hu/books?id=BM1ckQKCXP8C&sitesec=buy& hl=hu&source=gbs_vpt_read (vásárolható),