# SZAKDOLGOZAT

**Somodi Gábor Soma**
***2014***

Nyugat-magyarországi Egyetem
Geoinformatikai Kar

# Nyílt forráskódú rendszerek geodéziai felhasználási lehetőségei

SZAKDOLGOZAT

Konzulens:
Dr. Tóth Zoltán
Adjunktus

Készítette:
Somodi Gábor Soma
Nappali tagozat
Földmérő és földrendező
Mérnök Bsc Szak
Geoinformatikai szakirány

Székesfehérvár
*2014*

# NYILATKOZAT

Alulírott **Somodi Gábor Soma** (neptun kód: [redacted]) jelen nyilatkozat aláírásával kijelentem, hogy a Nyílt forráskódú rendszerek geodéziai felhasználási lehetőségei című

**szakdolgozat**

(a továbbiakban: dolgozat) **önálló munkám**, a dolgozat készítése során betartottam *a szerzői jogról szóló 1999. évi LXXVI. tv.* szabályait, valamint az egyetem által előírt, a dolgozat készítésére vonatkozó szabályokat, különösen a hivatkozások és idézések tekintetében[1].

Kijelentem továbbá, hogy a dolgozat készítése során az önálló munka kitétel tekintetében a konzulenst illetve a feladatot kiadó oktatót **nem tévesztettem meg.**

Jelen nyilatkozat aláírásával tudomásul veszem, hogy amennyiben bizonyítható, hogy a dolgozatot **nem magam készítettem,** vagy a dolgozattal kapcsolatban szerzői jogsértés ténye merül fel, a Nyugat-Magyarországi Egyetem **megtagadja a dolgozat befogadását és ellenem fegyelmi eljárást indíthat.**

A dolgozat befogadásának megtagadása és a fegyelmi eljárás indítása nem érinti a szerzői jogsértés miatti egyéb (polgári jogi, szabálysértési jogi, büntetőjogi) jogkövetkezményeket.

Székesfehérvár, 2014. május 26.

.....
Hallgató

---

1 1999. évi LXXVI. tv. 34. § **(1) A mű részletét - az átvevő mű jellege és célja által indokolt terjedelemben és az eredetihez híven - a forrás, valamint az ott megjelölt szerző megnevezésével bárki idézheti.**

36. § **(1) Nyilvánosan tartott előadások és más hasonló művek részletei, valamint politikai beszédek tájékoztatás céljára - a cél által indokolt terjedelemben - szabadon felhasználhatók. Ilyen felhasználás esetén a forrást - a szerző nevével együtt - fel kell tüntetni, hacsak ez lehetetlennek nem bizonyul.**

# Tartalomjegyzék

# Bevezetés

Szakmai tanulmányaim kezdetével egyre több időt fordítottam informatikai tudásom bővítésére, ezen belül is a szabadszoftverek megismerésére és a programozás elsajátítására.

A főiskolai évek alatt, az idő haladtával egyre több olyan szakmai tárgy volt, mely során szoftvereket kellett használnunk különböző feladatok elvégzéséhez. Ezen programok többsége tulajdonosi szoftver, azaz hosszútávú használatukért fizetni kell, ha korlátozások nélkül szeretnénk használni. Léteznek ingyenes változatai ezeknek a programoknak, azonban ezekből bizonyos funkciók tiltva vannak, vagy korlátozott ideig használhatóak.

Alternatívát jelentenek a szabadszoftverek, melyek egyik fő tulajdonsága, hogy szabadon használhatóak, másolhatók, terjeszthetőek és módosíthatóak, viszont ezek használatának elterjedtsége – összehasonlítva a tulajnodosi szoftverekkel szemben – elhanyagolható. Ez az arány igaz a geodéziai célú programokra is.

A fent leírtak miatt választottam szakdolgozati témámnak a nyílt forráskódú rendszerek geodéziai felhasználási lehetőségeit, melyben kitérek a szabadszoftverek fejlődésének fontosabb történeti eseményeire, a különböző licenctípusok ismertetésére. Az elterjedtebb, nagyobb közösség által használt geodéziai, térinformatikai célú szabadszoftverek rövid bemutatása után részletesen kitérek az általam végzett, geodéziai célú feladatok megoldására.

A programok, szkriptek írását a GNU Octave nevű magasszintű, interpretált nyelv használatával végeztem. A feladatokat Detrekői Ákos, Kiegyenlítő számítások című könyvéből választottam konzulensemmel egyeztetve. Célom a program alapvető funkcióinak bemutatása geodéziai feladatokon keresztül, illetve további felhasználási lehetőségek ismertetése.

# 1. Szabadfelhasználású szoftverek licenctípusai

A szoftvereket több szempont szerint is csoportosíthatjuk. Jelen esteben az egyik legfontosabb besorolás a licenctípus szerinti csoportosítás.

Alapvetően három nagyobb típust különböztetünk meg, ezek szerint lehetnek, tulajdonosi (proprietary) szoftverek, félszabad (semi-free) szoftverek és szabad (free) szoftverek. A fenti kategóriáktól elkülönítendő az un. „public domain“ szoftverttípus. Ezek nem állnak szerzői jogi szabályozás (copyright) alatt, szerzőjük ugyanis lemondott erről.

**A tulajdonosi szoftverek** csoportjába tartozik minden olyan program, mely használata, módosítása és terjesztése korlátozott. Ez a korlátozás nem feltétlen a szoftver ingyenességében mutatkozik meg. Az ide tartozó szoftverek nagy része kereskedelmi (commercial) program, de emellett lehetnek freeware és shareware verziók is.

A kereskedelmi programokat azzal a céllal hozzák létre, hogy kereskedelmi forgalomba hozzák őket, értéküket a tulajdonosok, szerzők szabják meg. Ezekre a programokra jellemző, hogy a felhasználó jogai pontosan be vannak határolva és a licenc szinte kizárólag a szerző érdekeit védi.

A freeware programok olyan programok, amelyeket szerzőjük ingyenessé nyilvánított. Ezek a programok szabadon terjeszthetők, és a felhasználásukra sincsen korlátozás. A szabad szoftverektől való eltérés a programok forráskódjában mutatkozik meg, ugyanis az nem ismerhető meg, és így a program nem is módosítható.

A shareware licenc alatt megjelentetett programok a freeware programokhoz hasonló felhasználási feltételekkel rendelkeznek. Ezek is ingyenesen beszerezhetők, és szabadon terjeszthetők. A fő különbség, hogy a shareware programok nem használhatók díjfizetés nélkül korlátlanul és tejleskörűen. A főbb korlátok általában a regisztrációhoz kötött és korlátozot idejű használatban mutatkoznak meg. Az időkorlát általában a program telepítésétől indul, és fix ideig (leggyakrabban 30 napig) tart, de konkrét naptári időponthoz is köthetik. A fenti típusok mindegyikébe besorolható például egy vírusírtó program, vagy akár egy CAD-szoftver.

Létezik kipróbálásra kiadott (trial), shareware verzióhoz hasonló szoftver. Ezek a programok azonban nem terjeszthetők szabadon. A limitált számmal kiadott (limited edition) szoftver korlátozása nem időhöz kötött, hanem a program működését korlátozzák, bizonyos szolgáltatásokat, funkciókat letiltanak, amíg meg nem vásároljuk a terméket.

A szoftvereket, ha ellátják reklámfelületekkel, akkor un. ad-powered szoftverekről

beszélünk. Ezek általában trial, shareware vagy freeware programok, melyeknél a bevétel a reklámokból adódik. Egy másik átmeneti típus a freeware és shareware között az un. postcard-ware (email-ware), amelyben a shareware programoknál megszokott korlátozás jelképes, s a program készítője annyit kér csupán a felhasználótól, hogy ha a termék megfelelt az elvárásainak, akkor azt jelezze a program készítőjének.

Az abandon-ware kategóriába olyan szoftverek tartoznak, melyek licencfeltételei megválzotak, készítőjük átsorolta (általában a kereskedelmi szoftver típusából valamely más kategóriába). Olyan szoftverek esetén jellemző az ilyen átsorolás, melyek területén a fejlődés, illetve fejlesztések gyors ütemben történnek. A felhasználó az átsorolás után teljes értékű, ingyenes termékhez jut hozzá, a forráskód is elérhetővé válik számára.

**A félszabad szoftverek** egy olyan átmeneti kategória a tulajdonosi és szabad szoftverek között, mely felhasználó személyétől, illetve a szoftver felhasználásának céljától teszi függővé annak státuszát. A szoftver felhasználása a magán-és üzleti célú használatban mutatkozik meg. Az oktatásban való felhasználást szinte mindig a kedvezőbb kategóriába sorolják. Ezek a programok az üzleti felhasználók számára kereskedelmi programokként, a magánfelhasználók számára freeware vagy szabad szoftverekként jelennek meg.

## 1.1. A szabad szoftver

Itt a "szabad" jelző nem azonos a freeware kifejezésben szereplő "free" szóval, amely "ingyenes"-ként fordítható.

A Free Software Foundation (FSF, szabad szoftver alapítvány) egy nonprofit szervezet, amelyet a szabad szoftver mozgalom, és azon belül is elsősorban a GNU projekt támogatására hozott létre Richard Stallman 1985. október 4-én.

A szabad szoftverlicencelési módot kidolgozó és népszerűsítő szervezet szerint a szabad szoftver felhasználójának a következő szabadságokkal kell rendelkeznie: [*http://fsf.hu*]

1. Lehetőség van a szoftver működésének szabad tanulmányozására és módosítására.
2. Szabadon terjeszthető, továbbadható.
3. Lehetőség van a szoftver továbbfejlesztésére és a fejlesztés közreadására.
4. A szoftver tanulmányozásának, módosításának, illetve továbbfejlesztésének előfeltétele a forráskód elérhetősége.

A szoftver bármilyen célra felhasználható. A program ingyenesen beszerezhető, általában letölthető az internetről, de bármilyen más terjesztési mód is szóba jöhet, például újságok CD-mellékeltén, vagy akár ismerőstől, az ő példányáról való másolat készítésével. Ez a

feltétel nem zárja ki azt, hogy a programot ellenérték fejében is terjesszék. Az ilyen módon terjesztett programok mellé várhatóan kapunk plusz szolgáltatásokat, dokumentációt, garanciát. A program terjesztőjének a terjesztés megkezdéséhez nincs szüksége a program készítőjének engedélyére. A terjesztéssel szemben két követelmény áll fenn:

1. Ha a forgalmazó csak a bináris kódot terjeszti, akkor a forráskódot a felhasználó kérésére minden esetben, ingyenesen annak rendelkezésére kell bocsátani.
2. A programmal a dokumentációját is terjeszteni kell.

Mivel a program forráskódja megismerhető, ezáltal a program módosítható, valamint ennek alapján új program (származék) hozható létre. A származék felhasználásának feltételeit a származék készítője határozza meg, és akár más felhasználási kategóriába is sorolhatja, pl. az üzleti szoftverek csoportjába. Ezen probléma kiküszölölése érdekében hozták létre a "copyleft" eljárást, amely a szabad szoftverek között újabb kategóriákat teremt.

A copyleft eljárásnál a felhasználási feltételeket úgy alakították ki, hogy az ilyen programokat bárki szabadon használhassa és módosíthassa. Tehát a módosító úgy is dönthet, hogy a program kikerülhet a szabad szoftverek köréből, az eredeti program készítőjének akarata ellenére. A copyleft eljárás ezen probléma megoldása érdekében a felhasználók jogait védi, azáltal, hogy a szabad szoftver alapján készített szoftver szerzőjét kötelezik arra, hogy a származékot is szabad szoftverként adja ki. A korlátozás a szerzői jogvédelem szokásos eszközeivel valósul meg, azaz a licencszerződésben jelenik meg.

A copyleft megvalósítására dolgozott ki a GNU project egy fix felhasználási feltételcsomagot, a GPL-t (General Public Licence). Ez egy általános licencszerződés, amelyet egy programkészítő felhasználhat az általa írt program felhasználási feltételeinek megállapításakor, azaz a programot a GNU General Public Licence feltételei alatt teszi elérhetővé. A GNU GPL nem tartalmaz korlátozásokat a szoftver használatára, csak a terjesztésre és a módosításra.

Érdekesség, hogy az FSF nem fogadja el hivatalosnak (sok más nyelvű fordítással együtt) a magyar nyelvűt. Ennek oka, hogy rendkívül költséges lenne számukra hitelesített szakfordítást rendelni. Az angol nyelvű dokumentációk a weboldalukról könnyedén elérhetőek.

A fenti terjesztési módnak köszönhetően a szabad szoftverek piaci részesedése jelentősen megnőtt, bebizonyosodott, hogy a "szabad szoftver" licencelési koncepció a piacon is életképes, ezt mutatják például a különböző Linux-disztribúciók kiadásával foglalkozó cégek

komoly sikerei (pl.: Linux Mint, Ubuntu, OpenSuse, Fedora stb.).

Az FSF mellett létezik egy másik csoport is, a nyílt forrás mozgalom, akik főként a nyílt forrású szoftverfejlesztés technológiai előnyei mellett érvelnek. A végeredmény szempontjából nincs jelentős különbség, csak az elvi hozzáállásban figyelhető meg eltérés. A nyílt forráskódú licenc olyan licenc, mely biztosítja a licencelt számítógépes program vagy szoftver forráskódjának nyitottságát. Egy licencet általában akkor tekintünk nyíltnak, ha azt az Open Source Initiative (OSI) jóváhagyja a Nyílt forrás definíció alapján. A közkincsként terjesztett programok is megfelelnek ezen követelményeknek, amennyiben a teljes forráskód elérhető, és így jogosultak az OSI "service mark" (szolgáltatási védjegy) használatára [*http://opensource.org/about*].

## 1.2. Magyarországi képviselet

### 1.2.1. Linux Ipari Szövetség (LIPSZ)

A Linux Ipari Szövetség 2004-ben jött létre a vállalati szintű nyílt forráskódú szoftverek, elsősorban a Linux és az erre épülő alkalmazások vállalati körben történő elterjesztésének és népszerűségének növelése érdekében.

### 1.2.2. FSF.hu Alapítvány

Az alapítvány célja a szabad szoftverek magyarországi honosítása, oktatása és népszerűsítése. Jelenlegi projektjeik közé tartozik Magyar Szabad Szoftver Tárház (MASZAT), mely célja a magyar fejlesztésű szabad szoftverek felmérése és minőségi értékelése. Jelenleg az mny2 projekt keretén belül A LIPSZ-el és ODFA-val közösen a minősítő rendszer kidolgozásán dolgoznak.

Egyéb projektjeik például a LibreOffice, Mozilla-család, Ubuntu és GNOME honosítása, magyar nyelvű dokumentáció készítése, illetve magyar nyelvi eszközök fejlesztése

### 1.2.3. Nyílt Dokumentumformátum Szövetség magyar tagozata

"A Nyílt Dokumentumformátum Szövetség 2006 márciusában jött létre az Egyesült Államokban azzal a céllal, hogy képzéssel, információ-átadással, szakmai tudásbázissal segítse a közszolgálatban tevékenykedőket és törvényalkotókat abban, hogy az exponenciálisan növekvő mennyiségű elektronikus dokumentumokat elsődlegesen OpenDocumentum (ODF) formátumban tárolják."

A Szövetség céljai, összhangban a nemzetközileg kinyilvánított célokkal:

1. Az ODF alkalmazásával kapcsolatos ismeretterjesztő és tudományos munka

támogatása.

2. Az ODF magyar nyelvű terminológiájának kidolgozása.
3. Az ODF alapjain, annak továbbfejlesztése érdekében kutatás-fejlesztés (K+F) és tudományos tevékenység folytatása.
4. Az ODF oktatása, az ezzel kapcsolatos képességfejlesztés.
5. A szabadon felhasználható informatikai szakirodalom magyar nyelvre fordítása.

### 1.2.4. GNU.hu

Az FSF.hu Alapítvány ezen az oldalon adja közre a GNU kiáltvány magyar fordítását, valamint az FSF által kiadott licencek (GPLv2, GPLv3, LGPLv2.1, LGPLv3, FDLv1.2) nem hivatalos magyar fordításait.

### 1.2.5. Szabad Szoftver Intézet (SzSzI)

A Szabad Szoftver Intézetet négy magánszemély alapította 2004-ben, azzal a céllal, hogy a szabad, nyílt forráskódú szoftverek használatát terjessze és oktassa.

Az Intézet elsődleges célja a szabad szoftverek használatának népszerűsítése, terjesztése, oktatása, valamint a szükséges szerződések birtokában, a világban elfogadott szabad szoftver tudást bizonyító vizsgák magyarországi szervezése, bonyolítása.

### 1.2.6. E-közigazgatási Szabad Szoftver Kompetencia Központ (EKOP-1.2.15)

Az E-közigazgatási Szabad Szoftver Kompetencia Központ célja több, szabad szoftverek bevezetésére irányuló pilot projekt szakmai támogatása. A Központ a pilot projektekről esettanulmányokat készít, amelyek segítenek a későbbi szabad szoftveres átállások megvalósítását [*http://szabadszoftver.kormany.hu*].

A projekt fontosnak tartja, hogy:

- jelentősen csökkenjen a közszférában a szoftver licencdíjak költsége,
- emelkedjen az állami informatikai szolgáltatások színvonala, tekintettel a rendszerek közötti interoperabilitás növekedésére,
- emelkedjen az állami informatikai szolgáltatások színvonala, tekintettel arra, hogy a licencköltségek csökkenésével elért megtakarításon magasabb szintű szolgáltatásokat lehet majd biztosítani,
- a szoftverjogok (használat, fejlesztés, továbbfejlesztés, stb.) birtoklásával a nemzeti adatvagyon felhasználása során megszüntethetőek legyenek a káros szállítói

függések.

A központ az alábbi tevékenységekkel foglalkozik:

- Hazai és nemzetközi tapasztalatok vizsgálata

A központ megvizsgálta a korábbi hazai és nemzetközi dokumentációkat, melyek a szabad szoftverekkel, azok jellemzőivel, bevezetésével, migrációjával, gazdasági és egyéb hatásaival foglalkoztak.

- Bevezetési és támogatási feladatok

A központ együttműködik több közintézménnyel, melyeknél pilot projekt keretén belül szabad szoftvereket vezetnek be, illetve korábban bevezetett szoftvereket bővítenek, oktatási-és segédanyagokat terjesztenek. A projektek elsődleges célja a szabad szoftverek közigazgatási használhatóságával kapcsolatos tapasztalatszerzés, ismeretgyűjtés.

- Oktatási anyagok készítése és oktatás

Felhasználók, oktatók és rendszergazdák oktatását is végzik szabad szoftver témákkal kapcsolatban. A felhasználói és az oktatói képzés főként a LibreOffice szoftverről, illetve használatáról szól. A központ által készített oktatási anyagok és az oktatásokról készült videók folyamatosan publikálásra kerülnek a weboldalon.

A központ által készített egyik oktatási anyag a LibreOfficehoz készített dokumentáció, mely *"Libreoffice Writer kalauz – Szövegszerkesztés stílusosan"* címmel jelent meg.

- Szabad megoldások számba vétele

A dokumentumgyűjtemény elkészítése során összegyűjtötték a kisebb-nagyobb cégek általános számítógépes szerver-igényének kielégítéséhez szükséges funkciókat, majd megvizsgálták, hogy az adott igényeket szabad szoftverekkel meg lehet-e oldani. A gyűjtemény folyamatosan bővül új dokumentumokkal.

- Tudástár létrehozása és karbantartása

A Tudástárban a kompetencia központ által a szabad licencekről, a szabad szoftverek hazai és nemzetközi felhasználásáról készített tanulmányok olvashatók.

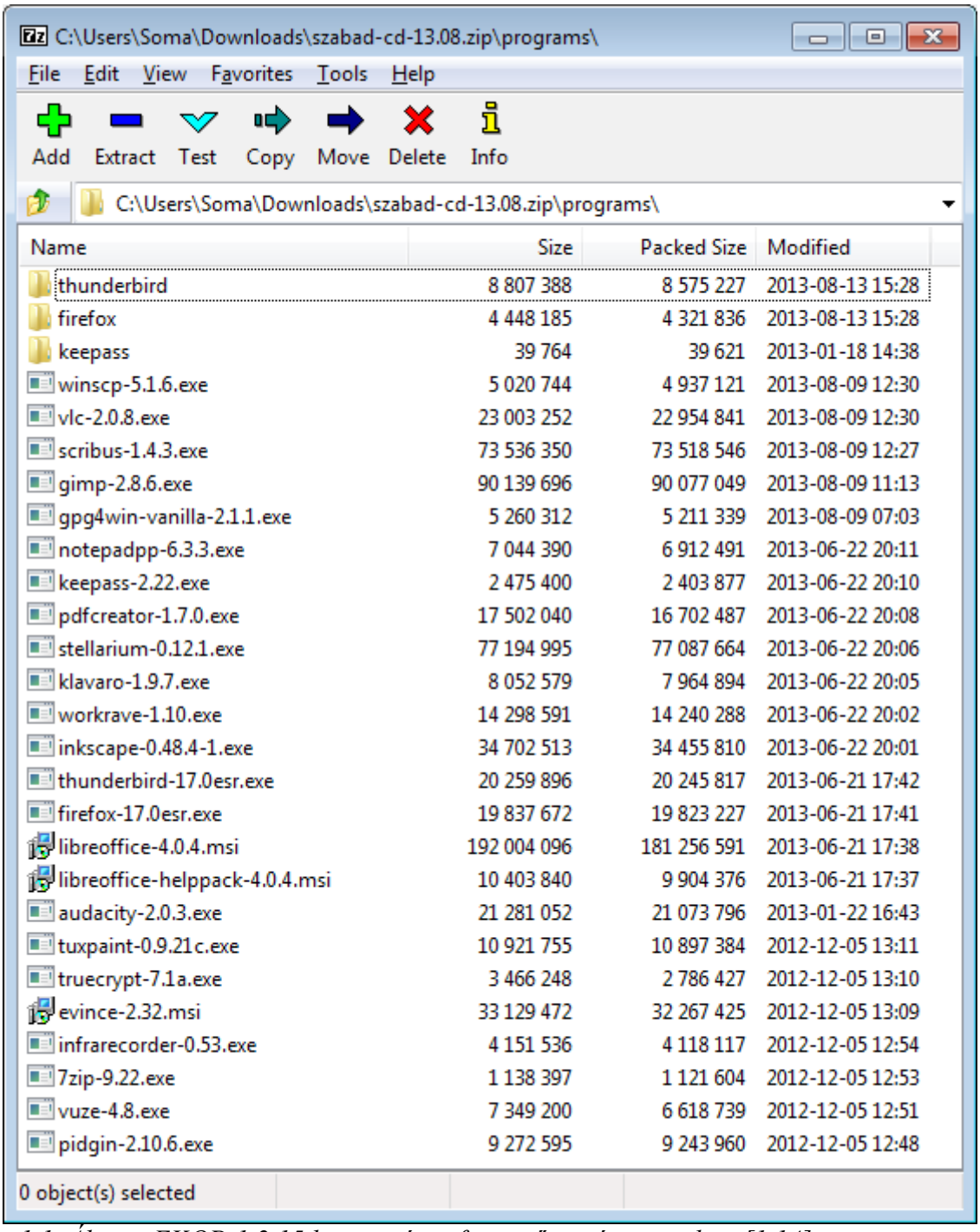
- Szoftverek honosítása

A kompetencia központ több fontos szabadszoftver-csomag honosításának karbantartását végzi, ezek közé tartoznak az FSF.hu alapítvány által fordított szabad szoftverek is.

- Bemutató szoftvergyűjtemény

Elérhető a kompetencia központ honlapjáról egy windowsos asztali szoftvereket

tartalmazó gyűjtemény, amelynek letöltésével kipróbálhatjuk, megismerhetjük a szabad szoftvereket, megnézhetjük tudásukat. A gyűjteményben többek között megtalálható a LibreOffice irodai programcsomag és az ehhez készült dokumentációk, a Mozilla Firefox böngésző és a Mozilla Thunderbird levelező, a GIMP és Inkscape képszerkesztők is. Több olyan szoftvert is tartalmaz a gyűjtemény, melyet a kompetencia központ honosított.




*1.1. Ábra - EKOP-1.2.15 bemutató szoftvergyűjtemény tartalma [1.14]*

# 2. Szabadszoftverek

Az alkalmazói szoftverek közül számos szabad szoftver lelhető fel irodai alkalmazások, üzleti alkalmazások, különböző tervezőrendszerek, kommunikációs szoftverek, hálózati alkalmazások és biztonsági programok körében. Ezek legfőbb előnye – a fentebb leírt nyílt megoldás mellett – a költségükben mutatkozik, ugyanis mindegyik kategóriában vannak ingyenesen használható programok, viszont a tulajdonosi szoftverekhez képest tudásuk általában kevesebb, fejlesztői csoportjuk kisebb, gyakran önkéntes alapon működik.

Az alábbi fejezetben ismertetem az elterjedtebb irodai programcsomagokat, képfeldolgozó szoftvereket, geodéziai és térinformatikai célú szabadszoftvereket.

## 2.1. Irodai alkalmazások

### 2.1.1. Apache OpenOffice, LibreOffice

Az Apache OpenOffice egy nyíltforráskódú, platformfüggetlen irodai szoftvercsomag. Az eredetileg a StarDivision által "StarOffice"-ként, később nyílt forrású termékként OpenOffice.org név alatt fejlesztett irodai programcsomag 2011-ben lépett be az Apache Incubator projektbe, majd 2012-ben léptették elő Apache felsőszintű projektté (Apache Top-level Project).

A Microsoft Office alternatívája, köszönhetően a hasonló felületnek és képességeknek, a fájlformátumok nagyfokú kompatibilitásának és nem utolsó sorban az ingyenességének. Az OpenOffice.org alapértelmezettként az ISO/IEC által szabványosított OpenDocument fájlformátumot használja, de megnyitható vele számos más irodai szoftvercsomag által használt formátum, köztük a Microsoft Office fájlformátumok. Az ODF szabvány (ISO/IEC DS 26300) és nyílt platform lehetővé teszi, hogy bármiképp is változzék a technológia, az elektronikus iratok teljes mértékben elérhetőek, szerkeszthetőek és menthetőek legyenek.

Forráskódja szabadon elérhető az Apache License 2.0 licenc alatt. Köszönhetően nyílt forráskódjának, számos operációs rendszerre elérhető, köztük a Microsoft Windows, Linux és Mac OS X rendszerekre. Az Apache Software Foundation 2014. április 17-én jelentette be, hogy eddig 100 millió példányban töltötték le az Apache OpenOffice-t.

A Sun 2010-ben történt felvásárlását követően az OpenOffice projekt legfőbb támogatója az Oracle lett. A Sun felvásárlása körüli bizonytalanság hatására az OpenOffice projekt tagjai létrehozták a The Document Foundation alapítványt, ami a LibreOffice név alatt vitte tovább az OpenOffice leágazását. 2010-ben az FSF.hu Alapítvány is csatlakozott a Document Foundation alapítványhoz. 2011-ben elkészülnek a libreoffice.hu és a hu.libreoffice.org

honlapok, illetve megjelent a magyar nyelvű LibreOffice 3.3.

A LibreOffice és OpenOffice hat alkalmazásból áll: Writer (szövegszerkesztő), Calc (táblázatkezelő), Impress (bemutatókészítő), Draw (rajzoló), Math (képletszerkesztő) és Base (adatbázis-kezelő).

### 2.1.2. Google Dokumentumok (Google Docs)

A Google dokumentumok egy webalapú irodai alkalmazáscsomag a Google-től, mely táblázatkezelőt, szövegszerkesztőt és prezentációkészítőt is tartalmaz. Ajax és XML használatán alapul, a Web 2.0 típuspéldájaként. Dokumentumokat egy Gmail postafiók birtokában tudunk létrehozni, módosítani és menteni. A fájlok mérete korlátozott, szöveges dokumentumok maximum 1,024,000 karaktert tartalmazhatnak, formázástól függetlenül. Ez körülbelül 10 MB. A táblázat maximális mérete 400,000 cella, munkalaponként maximum 256 oszloppal. Prezentáció maximális mérete 50 MB lehet, ez körülbelül 200 diát jelent.

A létrehozott fájlokat lemethetjük meghajtónkra, vagy Google Driveon is tárolhatjuk, mely szintén egy ingyenes szolgáltatás. Funkcionalitásában nem olyan erős, mint például a LibreOffice, de átlagfelhasználásra elegendő. Előnye, hogy böngészőben fut, telepíteni nem kell és egyszerre több felhasználó is szerkesztheti a dokumentumokat.

### 2.1.3. Office Online

Hasonlóan a Google Dokumentumokhoz, a Microsoftnak is elérhető egy ingyenesen elérhető Office változata, melyet szintén böngészőből használhatunk Microsoft-fiókunkkal. Létrehozhatunk dokumentumokat, számolótáblákat és bemutatókat online módon vagy az Office asztali számítógépen telepített verziójával, lehetőségünk van a OneDrive online tárterületére menteni a fájlokat, illetve a Google Dokumentumokhoz hasonlóan megoszthatjuk a fájlokat másokkal a valós idejű együttműködéshez.

### 2.1.4. TeX, LaTeX

A TeX nyomdai minőségű dokumentumok előállítására alkalmas szedő-és tördelőprogram, melyet Donald E. Knuth fejlesztett ki. Általános vélemény szerint a TeX a legmegfelelőbb módja a matematikai képletek szedésének. A TeX-ben makrók is írhatók, ez viszont lehetővé teszi, hogy a TeX-re egy magasabb szintű, barátságosabb programnyelv épüljön.

A LaTeX egy TeX-en alapuló szövegformázó rendszer, amely nagyon alkalmas olyan elektronikus dokumentumok, szakdolgozatok, tudományos cikkek írására, amelyek sok

képletet tartalmaznak. A LaTeX alkotója Leslie Lamport. Az előzőekben ismertetett alkalmazásokhoz képest eltérés, hogy itt nem azt kapjuk végeredményül, amit látunk. *What You See Is What You Get* (WYSIWYG), azaz *"Azt látod, amit kapsz"* szemlélet szerint működnek a LibreWriter és OpenOffice termékek. Ezzel ellentétben a LaTex esetén a szemléletmódszer az ún. *What You See Is What You Mean* (WYSIWYM), *"Azt látod, amit kapni akarsz"*. A WYSIWYM szemlélet a szöveg szerkezetére fektet nagyobb hangsúlyt. A szerkezetet és formázást élesen megkülönbözteti.

## 2.2. Képfeldolgozó szoftverek

### 2.2.1. GIMP

A GIMP (GNU Image Manipulation Program) egy rendkívül nagytudású pixelgrafikus képszerkesztő program, amely elérhető Windows, Linux és Mac OS X operációs rendszereken egyaránt.

A projektet 1995-ben Spencer Kimball és Peter Mattis kezdte, jelenleg önkéntesek csoportja tartja karban és fejleszti. A program GPL licenc alatt szabadon használható.

### 2.2.2. InkScape

Az Inkscape nyílt forráskódú vektorgrafikus szerkesztő. Célja teljesen betartani az XML, az SVG és a CSS szabványait a folyamatos fejlődés mellett. Keresztplatformos alkalmazás, amely fut Microsoft Windows, Mac OS X és Linux operációs rendszereken.

## 2.3. Geodéziai és térinformatikai célú szabadszoftverek

A technológiai fejlődésnek köszönhetően szakmánkban is teret hódítottak a szoftverek, mely a műszerekből való adatkiolvasástól, a geodéziai számításokon át, hálózatkiegyenlítéssel, transzformációval, konvertálással, vektoros szerkesztéssel is támogatta a felhasználót, feladattól függően. Ezen megoldások gyorsították, segítették az adatgyűjtést, feldolgozást és kiértékelést.

Jelenleg a használatban lévő programok főként tulajdonosi szoftverek, így azok teljeskörű használatáért fizetni kell. A hazai piacon a legelterjedtebbnek mondható CAD szoftverek az ITR, MicroStation és az AutoCAD, irodai alkalmazás és operációs rendszer tekintetében pedig a Microsoft termékei terjedtek el.

### 2.3.1. FreeTR

A FreeTR egy ingyenes térképszerkesztő rendszer, melynek segítségével létrehozhatunk, szerkeszthetünk és konvertálhatunk digitális térképeket. Rendelkezik DXF, DATR, ki-, és

bemenettel, alapértelmezetten pedig saját adatformátumában, egy *.ftr kiterjesztésű fájlban tárolja az adatokat.

### 2.3.2. QCAD

A QCAD egy számítógépes alkalmazás kétdimenziós tervrajzok készítéséhez. Segítségével műszaki rajzokat készíthetünk. Az alkalmazás Microsoft Windows, Mac OS X és Linux operációs rendszereken érhető el, forráskódja a hivatalos honlapról letölthető. A jelenlegi kiadás GPLv3 licenc alatt van.

### 2.3.3. QGIS

A QGIS egy nyíltforrású térinformatikai program, mely Linux, Android, Unix, Mac OSX és Windows rendszereken futtatható. Jelenleg 2.2-es verziónál tart a fejlesztés, GNU GPL licenc alatt. A QGIS többféle vektoros, raszteres és adatbázis formátumot támogat.

A fent említett programok az utóbbi pár év során látványos fejlődésen mentek keresztül, viszont fizetős vetélytársaiktól funkcionalitásban elmaradottnak számítanak. Magyar dokumentáció a jelenlegi (és pár előző) verzióhoz nem érhető el, csak a FreeTR-hez, magyar fejlesztés lévén.

# 3. GNU Octave

A GNU Octave egy magasszintű nyelv, mely elsősorban numerikus számításokra használható. Alapesetben parancssoros kezelőfelületet szolgáltat, lineáris és nemlineáris problémák numerikus megoldásához, illetve egyéb numerikus kísérletek véghezviteléhez, ami a Matlab-bal nagyrészt kompatibilis. Ugyanakkor felhasználható script-nyelvként is. A szoftvert elérhető Linux, Windows és OS X operációs rendszerekre, illetve bárki szabadon terjesztheti, és/vagy módosíthatja a GNU GPL-ben foglaltak szerint.

*3.1. Ábra - A GNU Octave logója [1.11]*

A GNU Octave alapjait 1988 környékén fektették le. Egy kísérő szoftvernek szánták egy tankönyv mellé, amely kémiai reakciók tervezésével és analízisével foglalkozik. A fejlesztés 1992 tavaszán kezdődött, az első tesztváltozat egy év múlva készült el, majd az 1.0-ás verziót 1994 februárjában adták ki.

A szakdolgozatban leírt feladatokhoz a GNU Octave 3.8.1 verzióját használtam Microsoft Windows 7 operációs rendszeren. Alapesetben az Octave parancssoros felülettel indul, a grafikus felhasználói felület (graphical user interface, GUI) a 4.0.x verzióban lesz alapértelmezett. A 3.8.0 verzió óta Windows rendszeren is elérhető a beépített GUI [*http://www.gnu.org/software/octave*].

## 3.1. Adattípusok

Az Octave beépített adattípusokkal rendelkezik, de lehetőségünk van új speciális adattípust definiálni. Octave-ban nem kell megadni a változók típusát. A program futása során, a nekik megfeleltetett konstans objektum típusa határozza meg. Egy programon belül ugyanaz a változó (ugyanolyan névvel) akár többféle típusú is lehet, mindig az utolsó neki megfeleltetett érték típusa határozza meg az aktuális típusát. Typeinfo beírásával a felhasználható adattípusok listáját kapjuk meg egy tömbben, mely jelenleg 54 adattípust tartalmaz. Az alábbiakban a beépített adattípusokat fogom részletezni, fontosabb tulajdonságaikat kiemelve. Zárójelben a konstansok, speciális értékek, illetve egyéb funkciókhoz tartozó Octave parancsot fogom megemlíteni. Példák, részletes leírás, egyéb segítség megtalálható az Octave dokumentációjában, illetve az interneten, ezért az alapvető példákat nem írom le, csak a megoldott geodéziai feladat forráskódját.

### 3.1.1. Numerikus objektumok

A numerikus objektumok a valós és komplex számok, vektorok, valamint a mátrixok. Minden beépített numerikus adattípus kétszeres pontosságú számként van eltárolva. Azon rendszereken, amelyek az IEEE lebegőpontos formátumot használják, 2.2251e-308 (realmin) és 1.7977e+308 (realmax) közötti értékek tárolhatóak, a relatív pontosság körülbelül 2.2204e-16 (eps). A mátrix objektumok bármekkora méretűek lehetnek és dinamikusan átméretezhetőek.

Egy numerikus konstans legegyszerűbb formája a skalár, amely lehet egész, tizedes tört, exponenciális alakban írt szám vagy komplex szám. Nem megfelelő formai bevitel esetén a fordító jelzi a hibát és annak helyét.

Mátrix definiálásánál a rendszer automatikusan eldönti a mátrix méretét, azt nem kell megadni. A mátrixot szögletes zárójelben definiáljuk, a sorokat szóközzel vagy vesszővel, az oszlopokat pedig pontosvesszővel elválasztva. Természetesen a nem megfelelő mátrixelrendezést a rendszer hibaüzenettel jelzi a felhasználó számára. Beépített fügvényekkel speciális mátrixokat hozhatunk létre. Ilyen például a nullmátrix (zeros), egységmátrix (eye), vagy csupa egyesekből álló mátrix (ones).

Vektorokat sorozatként tudunk definiálni. Megadási módja történhet a mátrixhoz hasonlóan, szögletes zárójelben a számokat vesszővel vagy szóközzel elválasztva. Másik módszer három szám kettősponttal való elválasztása. Ekkor megadjuk, hogy mekkora értéktől mekkora lépésközzel meddig tartson a sorozatunk. Megadhatjuk egy intervallum határai mellé a sorozat elemszámát is (linspace), ekkor a két határ között a megadott hosszú sorozatot kapjuk eredményül.

Két beépített logikai változó van, ezek a false, és a true. Ezek a logikai változók implicit konvertálódnak skalárrá, ha ez szükséges - a false-ból 0, a true-ból 1 lesz. Logikai változókat felhasználhatunk mátrixok feltöltésére is (false és true parancsok, megadva a mátrixot).

Beépített konstans a hiányzó érték (NA). Két hiányzó érték nem egyenlő, és nem hasonlítható össze. NaN jelöli a "nem számot" (Not a Number). Két NaN nem egyenlő, és nem hasonlítható össze. NaN az értéke például a 0/0, az Inf-Inf, valamint a NaN-nal végzett műveleteknek egyaránt. Szintén beépített konstans a végtelen (Inf, -Inf). Két végtelen szám egyenlő, és minden más értéknél nagyobb, kivéve az NA-t, és a NaN-t.

### 3.1.2. String objektumok

Octave-ban a stringek idézőjelek (") vagy aposztróf jelek (') közé írt karaktersorozatokból állnak. Az Octave a stringeket karaktertömbökként tárolja, így minden indexelő művelet,

amelyeket mátrixok esetében használhatunk, stringeknél is alkalmazhatóak. A stringbe írt idézőjel használatát a "\" fordított perjel használata teszi lehetővé. A szekvenciák hasonlóak a C nyelvben használtakhoz, Octave-ban többet is tudunk használni.

### 3.1.3. Adatsturktúrák

Az Octave-ban lehetőség van arra, hogy az adatokat adatstruktúrákba szervezzük. A jelenlegi implementáció egy asszociatív tömböt használ. A struktúrák elemei bármilyen típusúak lehetnek, illetve egymásba ágyazhatóak.

### 3.1.4. Listák, cella tömbök

Az Octave két különböző módon támogatja az adattípusok tárolását ugyanabban a változóban. Az egyik a lista (vagy vesszővel elválasztott lista), mely az összes Octave függvény értéktípusa. A cella tömb pedig egy olyan tömb amelynek celláihoz különböző típusú és méretű elemeket rendelhetünk. Kezelésük hasonlóan történik a mátrixokéhoz, annyi különbséggel, hogy hivatkozásoknál a kapcsos zárójelet kell használni.

## 3.2. A beépített típusok műveletei

Numerikus objektum esetén az alábbi aritmetikai, összehasonlító, logikai műveletek és index-és értékadó kifejezések használhatóak:

Mátrix esetén azonos dimenzió szükséges szinte mindegyik művelet esetén.

- X + Y = Összeadás.
- X – Y = Kivonás.
- X * Y = Szorzás.
- X .* Y = Elemenkénti szorzás.
- X / Y = Jobb oldali (mátrix)osztás.
- X ./ Y = Elemenkénti jobb oldali osztás.
- X \ Y = Bal oldali (mátrix)osztás.
- X .\ Y = Elemenkénti bal oldali osztás. Az y minden elemét elosztja x megfelelő elemével.
- X ^ Y vagy x**y = Hatványozás.
- X .^ Y = Elemenkénti hatványozás.
- X' = Transzponált konjugáltja. Valós mátrix esetében ez megegyezik a

transzponálttal.

- X.' = Transzponált.

Ezek mellett használhatóak a C-ben használt formulák is:

- X++, ++X, X--, --X = Inkrementálás. A változó értékének 1-el való növelése/csökkentése.
- X^=3, X+=2, X-=1

Összehasonlító operátor használata esetén skalárt és mátrixot is összehasonlíthatunk, ekkor az eredmény egy logikai mátrix lesz, mely mérete az eredeti mátrix méretével egyező.

- X < Y = Igaz, ha X kisebb Y-nál.
- X <= Y = Igaz, ha X kisebb, vagy egyenlő mint Y.
- X == Y = Igaz, ha X és Y egyenlőek.
- X >= Y = Igaz, ha X nagyobb, vagy egyenlő mint Y.
- X > Y = Igaz, ha X nagyobb, mind Y.
- X != Y vagy X~=Y vagy X<>Y = Igaz, ha X nem egyenlő Y-nal.

A logikai műveleteknek két fajtája van. Vannak egyszerű logikai műveletek:

- X && Y = logikai és
- X || Y = logikai vagy

A másik csoportba tartoznak az elemenkénti logikai műveletek:

- X & Y = logikai és
- X | Y = logikai vagy
- ! X vagy ~ X = logikai nem
- Xor(X, Y) = logikai kizáró vagy

Ezeknek a műveleteknek az operandusai skalárok és mátrixok lehetnek. Ha mindkét operandus mátrix, akkor a dimenzióiknak meg kell egyezniük.

Egy mátrix vagy vektor elemeit index kifejezésekkel érhetjük el. Az indexek lehetnek skalárok, vektorok, sorozatok, vagy a teljes sort vagy oszlopot jelentő kettőspont operátor.

Az értékadással egy változónak új értéket adunk. Az x=42 kifejezésben x változóhoz 42-őt rendelünk hozzá, bármi is volt az értéke azelőtt. Az egyenlőség jelet értékadás opererátornak nevezzük.

A fenti műveletek mellett számos beépített függvény segíti munkánkat, mely során vizsgálhatjuk a numerikus, string és egyéb objektumok tulajdonságait. Ezen függvények az Octave dokumentációjában megtalálhatóak, leírással és példákkal együtt.

## 3.3. Vezérlési szerkezetek

Vezérlési szerkezetnek nevezünk egy olyan programnyelvi konstrukciót, amely az összetevői végrehajtási sorrendjét adja meg. Octave-ban a következő vezérlési szerkezeteket használhatjuk: if, switch, while, do-until, for, break, continue, unwind_protect és try szerkezet.

## 3.4. Függvények

Az Octave-ban lehetőség van függvények írására, amelyeket definiálhatunk az Octave parancssorán, szkirptként, vagy betölthetünk meglévő fájlokat is. Octave-ban lehetőség van olyan függvény definiálására, amelynek több visszatérési értéke is van. A későbbi példák során ezt a típusú megvalósítást láthatjuk majd.

Az Octave lehetőséget ad függvények szabványos dokumentálására. A függvényt tartalmazó fájl elején "##" (vagy "%%") kezdetű sorokban kommentként a szerzői jog szerepelhet, majd egy üres sor után hasonló módon a függvény dokumentációja. Ezek után következhet még több sor fejléc, ahol a fejlécnév lehet:

- A szerző neve.
- A neve és elérhetősége annak, aki az adott függvényért felel.
- Akit probléma esetén keresni lehet.
- A fájl létrehozásának dátuma.
- Verziószám.
- Függvény funkciójához kapcsolódó kulcsszavak.

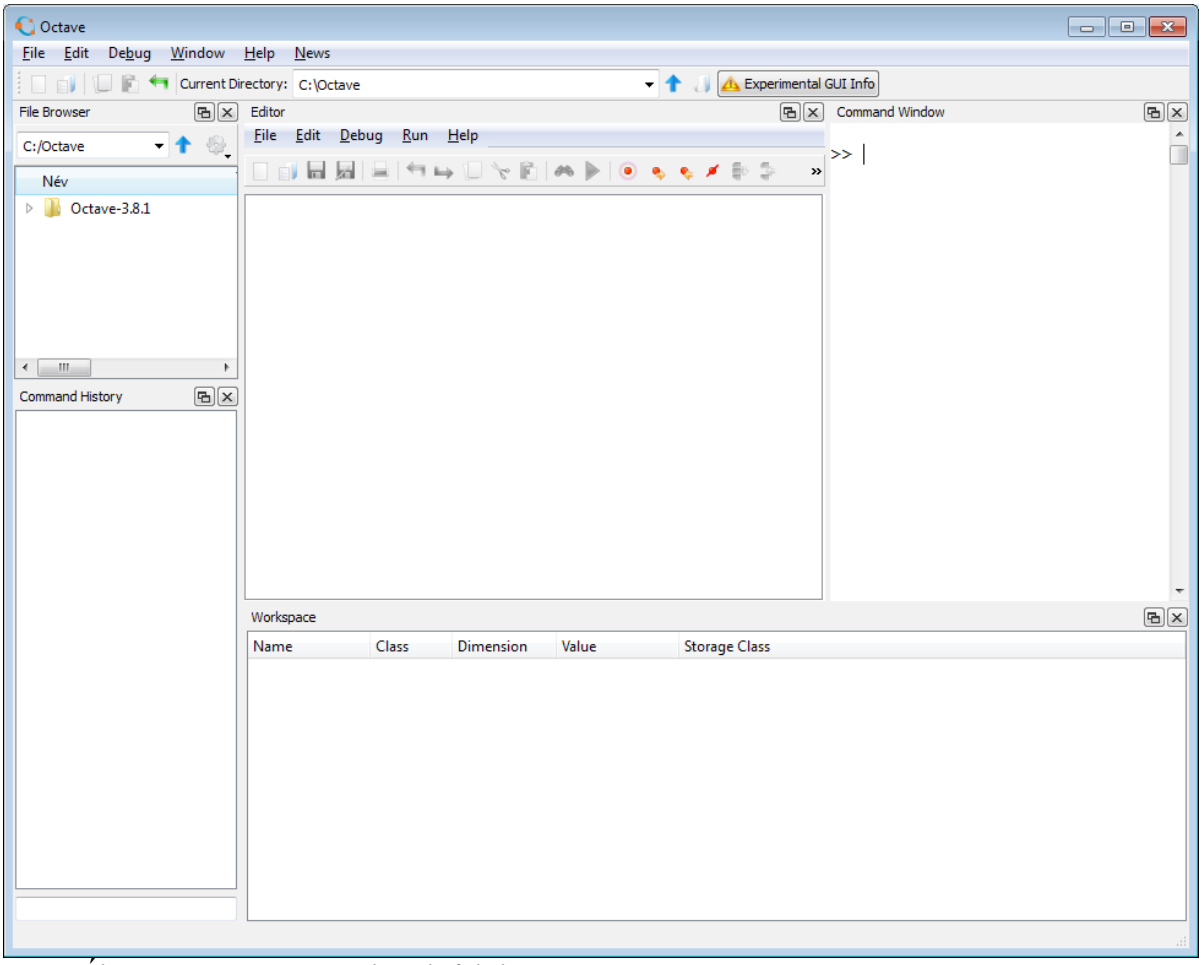
Az ilyen dokumentáció a beépített függvények dokumentációjához hasonlóan elérhető a help paranccsal.

## 3.5. A program felépítése

A szoftver telepítését követően elindíthatjuk a programot, mely grafikus felhasználói felülete az alábbi részekből épül fel:

- File browser
- Command history

- Workspace
- Command Window
- Editor

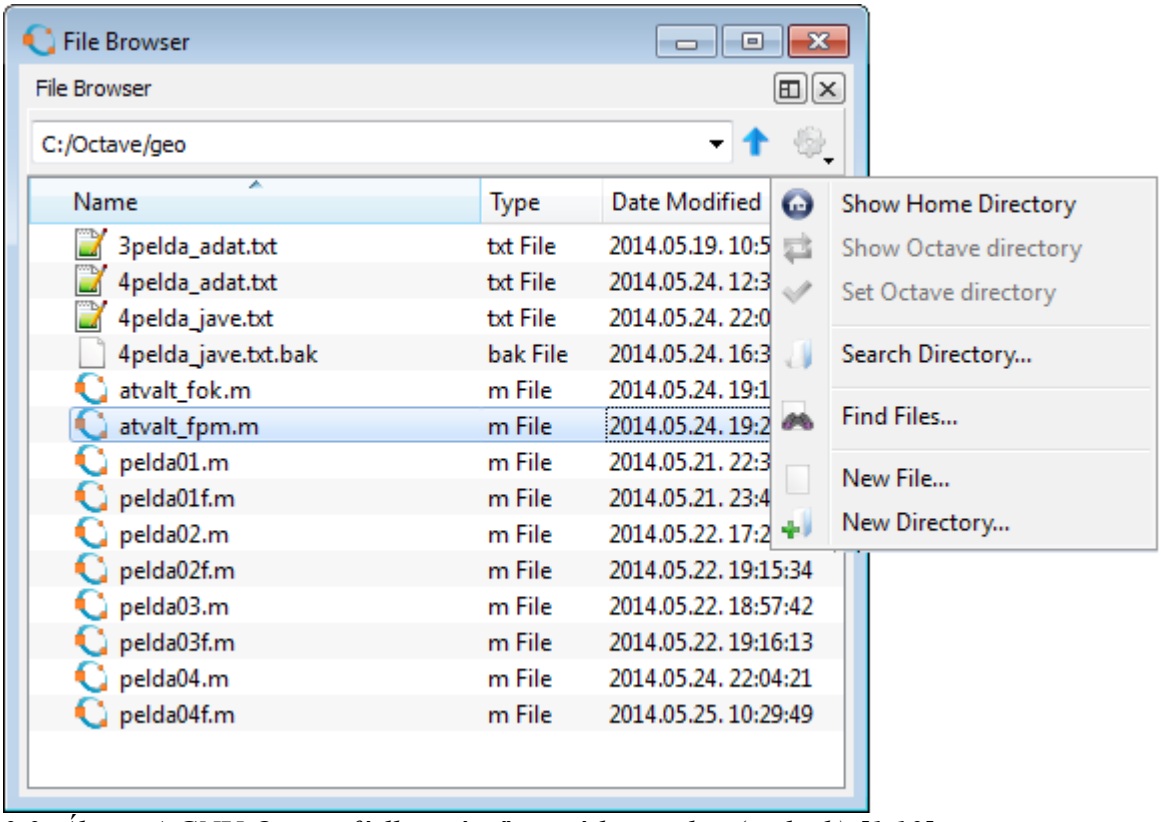



*3.2. Ábra - A GNU Octave kezelőfelülete [1.12]*

A különböző részek szabadon átméretezhetőek és áthelyezhetőek, illetve kikapcsolhatóak, dokkolhatóak.

### 3.5.1. File browser

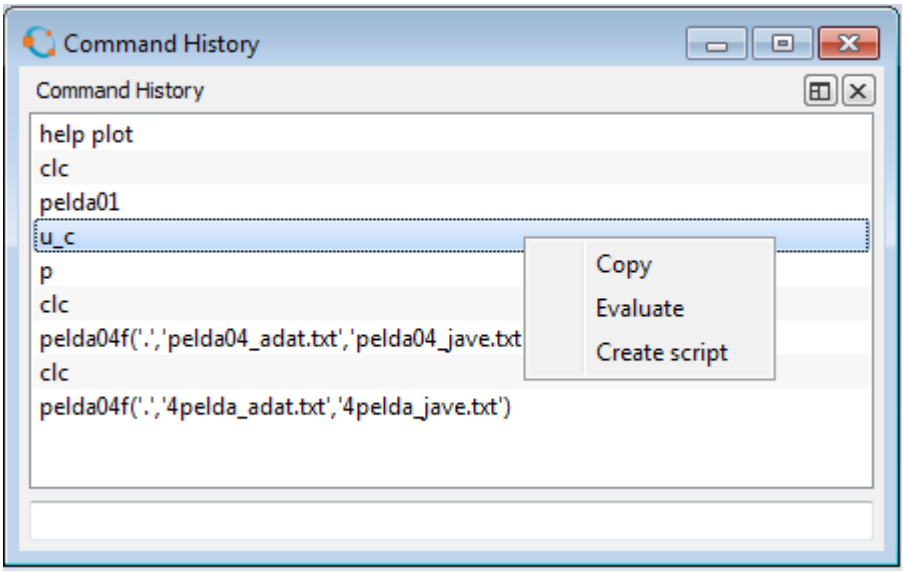
A fájlböngésző segítségével könnyel elérhetjük könyvtárainkat, Octave kompatibilis fájljainkat. A keresőben megadott mappán belül keresni is tudunk fájlnév alapján. A "csavar ikon"-ra kattintva legördülő menün belül létrehozhatunk és kereshetünk fájlokat, mappákat. Új fájl vagy mappa létrehozása mindig az aktuális könyvtáron belül történik.




*3.3. Ábra - A GNU Octave fájlböngészője, szétkapcsolva (undock) [1.10]*

### 3.5.2. Command history

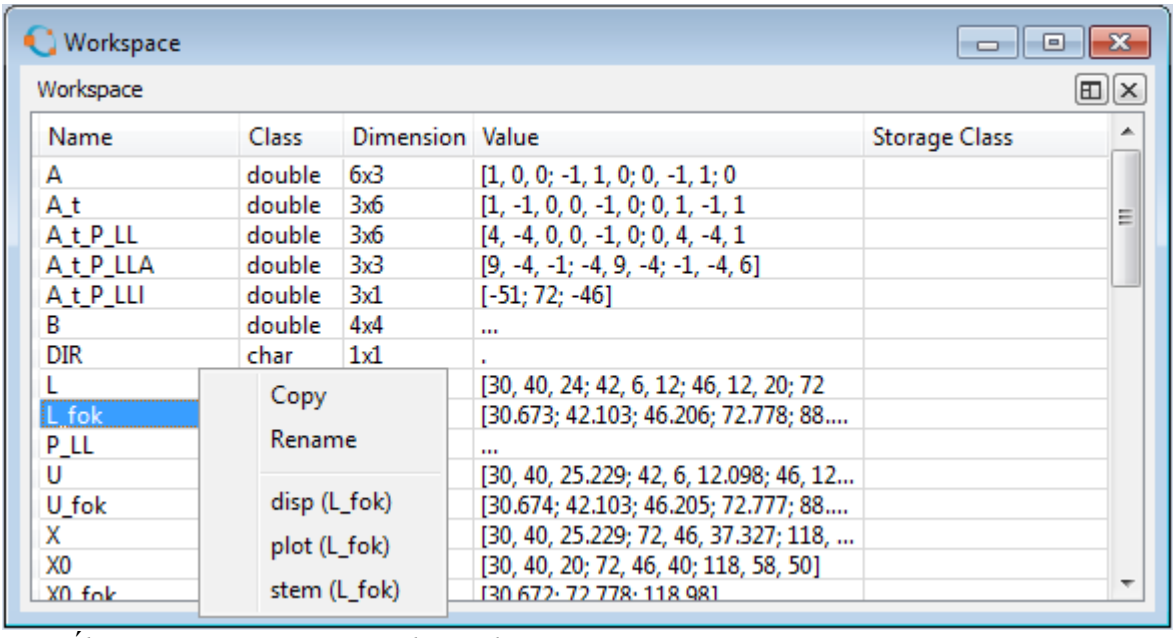
A Command Window-ba írt parancsok és az Editorban lefuttatott szkriptek jelennek meg az előzményeknél. Egy előzőleg beírt parancsot innen tudunk másolni, újra lefuttatni.




*3.4. Ábra - A GNU Octave előzmény ablaka [1.7]*

### 3.5.3. Workspace

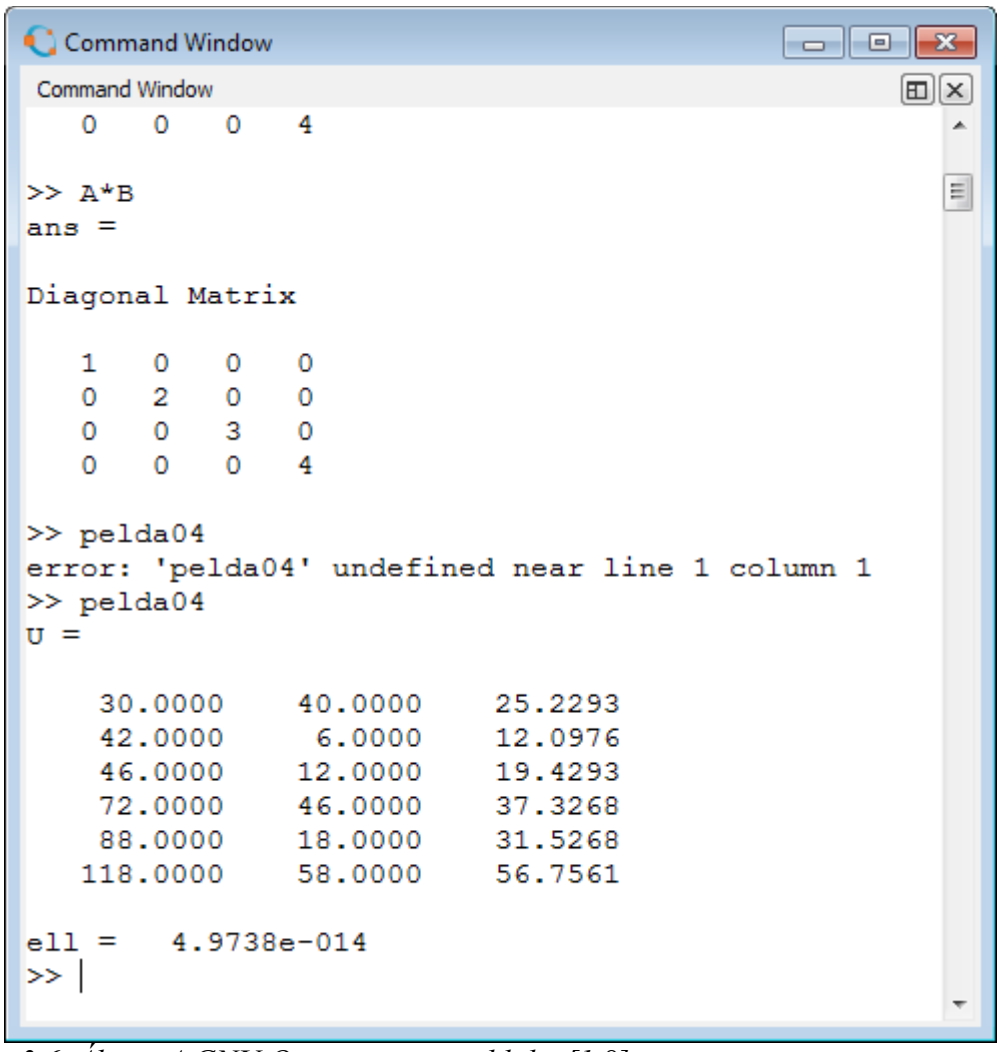
Minden számítási eredményünk a munkatelületen érhető el. Ha nem használunk változókat a számítások során, akkor a legutóbbi számítás eredménye az *ans* nevű változóba kerül. Információt kapunk a név mellett még az osztály típusáról, dimenzióról, a tárolt értékről és a tárolási osztályról.




*3.5. Ábra - A GNU Octave munkaterülete [1.13]*

### 3.5.4. Command Window

Alapesetben a számításokat a parancsablakban végezzük, a prompt (>>) után írhatjuk be az utasításokat. Különböző függvényeket hívhatunk meg, illetve a függvény-és egyéb parancsok használatáról itt kérhetünk segítséget a *help* paranccsal.




*3.6. Ábra - A GNU Octave parancsablaka [1.8]*

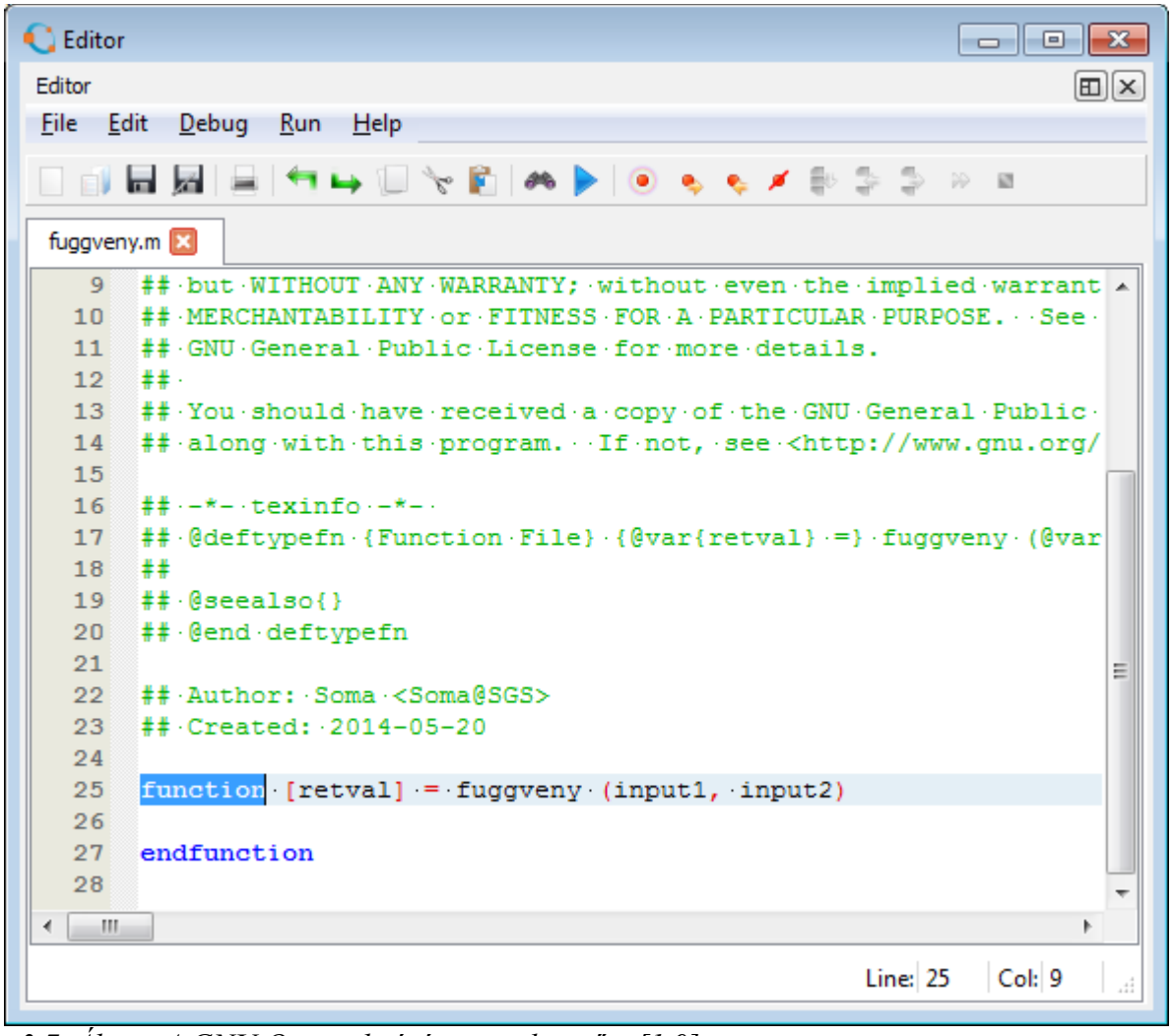
### 3.5.5. Editor

Ha sok számítást kell egymás után elvégeznünk, és nem vagyunk kíváncsiak a részeredményekre, akkor nem szükséges a parancsablakban minden egyes utasítást egymás után begépelni, hanem érdemes m-fájlt készíteni. Ilyen fájlokat az Octave beépített szerkesztőjében, vagy akár egy egyszerű szövegszerkesztőben is létre tudunk hozni.

Az adott .m állomány nevét a parancsablakban utasításként kiadva az Octave a megfelelő szöveges állomány utasításait egymás után értelmezi és végrehajtja. Az .m állományokban lehetőség van vezérlési szerkezetek létrehozására, más .m kiterjesztésű fájlok meghívására. Hasonlóan a parancsablakhoz, itt is a pontosvessző szolgál az utasítások szétválasztására.

Ha a .m kiterjesztésű fájlunk nem az aktuális könyvtárban, vagy az Octave egy olyan könyvtárában van, melyben alapértelmezetten megkeresné, akkor azt futtatás előtt hozzá kell adnunk az átvizsgálandó könyvtárak közé. Ezt az Octave alapértelmezetten felajánlja

számunkra első futtatáskor.




*3.7. Ábra - A GNU Octave beépített szerkesztője [1.9]*

A fájl menüben többek között létrehozhatunk új m-fájlokat és új függvényeket. A 3.7 ábrán látható egy ezen módszerrel létrehozott függvény. Ilyenkor a program automatikusan beszúrja a szerzői jogokat és egy szabvány dokumentációt a függvény leírására. Ha az első összefüggő kommentezett rész nem tartalmazza a *Copyright* szót, akkor azt tekinti az Octave a függvény dokumentációjának.

### 3.5.6. Súgó

A funkciók és függvények teljeskörű megismeréséhez elengedhetetlen a hivatalos dokumentáció használata. Az Octave-nak a telepített kézikönyve segítséget nyújt a kezdeti lépésekben és komplexebb megoldásokban egyaránt, a beépített műveleteket példákon keresztül mutatja be. A Command Windowból a *help "függvénynév"* használatával is elérhető a funkciók és függvények dokumentációja. A beépített és a letölthető kézikönyv magyar nyelven jelenleg nem elérhető.

# 4. Geodéziai számítások Octave használatával

Az Octave program geodéziai számítási lehetőségeit kisebb feladatokon keresztül mutatom be [Detrekői, 1991.]. Főbb célom, hogy az elkészített megoldások minél általánosabbak legyenek, ne csak az alkalmazott adatokkal legyen használható egy szkript, hanem az adott feladattípus esetén bármilyen adattal.

## 4.1. Normális eloszlás

**1. Példa:** Tekintsük $\xi$ normális eloszlású valószínűségi változónak valamely hosszmérés eredményét. Tételezzük fel, hogy várható értéke, $a$=80,200 m és szórása, $\sigma$=0,04 m ismert. Mi a valószínűsége annak, hogy egy mérés eredménye a 80,140 m és a 80,220 m értékek közé esik?

A mérések eredményeinek feldolgozásakor általában azzal a feltételezéssel élünk, hogy a mérési eredmények normális eloszlásúak (a tényleges elvégzett mérések is ezt mutatják). Az *f(x)* sűrűségfüggvény az eloszlásfüggvény derivált függvénye.

A $\xi$ valószínűségi változót normális eloszlásúnak nevezzük, ha sűrűségfüggvénye

$$f(x)=\frac{1}{\sigma\sqrt{2\pi}}e^{\frac{-(t-a)^2}{2\sigma^2}} \quad , \tag{4.1}$$

ahol $a \in R$, $\sigma > 0$. Jelölés: $\xi \sim N(a, \sigma)$ vagy $F(x; a, \sigma)$.

A várható érték és medián jele $a$, a szórás jele $\sigma$. A (4.1) függvényből látható, hogy az $a$ várható értéke és a $\sigma$ szórás egyértelműen meghatározza a normális eloszlást. Az *f(x)* függvény szimmetrikus az $a$ értékre. Az $N(a, \sigma)$ eloszlás sűrűségfüggvényének képe a Gauss-görbe vagy más néven haranggörbe, melynek helyzetét és alakját az $a$ várható érték és $\sigma$ szórás határozza meg. Kisebb szórású eloszlás Gauss-görbéje meredekebb, nagyobb szórásúé laposabb lesz.

Octave használatával a feladat könnyen megoldható. Az értékadás után a változó előjelét vizsgáltam, majd az alábbi, standardizált normális eloszlás eloszlásfüggvényére igaz összefüggés alapján változott a végeredmény:

$$\Phi(-u)=1-\Phi(u) \tag{4.2}$$

Ezután az Octave beépített *stdnormal_cdf* függvényét használtam, mely megadja, mekkora valószínűséggel esik a *c, d* intervallumba a $\xi$ valószínűségi változó.

A programkód: [Mellékletek: CD, *4.3*]

```
%Adatok:
a = 80.200;
sigma = 0.04;
c = 80.140;
d = 80.220;
u_c = (c-a)/sigma
u_d = (d-a)/sigma
%Képletek:
if (u_c < 0) %2.63 képlet alapján átalakítás [Detrekői, 1991.]
fi_c = 1 - stdnormal_cdf(abs(u_c));
else
fi_c = stdnormal_cdf(u_c);
endif

if (u_d < 0)
fi_d = 1 - stdnormal_cdf(abs(u_d));
else
fi_d = stdnormal_cdf(u_d);
endif
p = fi_d - fi_c %Végeredmény
```

Eredményül a $p = 0.62466$ valószínűséget kaptuk.

### 4.1.1. Függvény fájl létrehozása

A fenti szkript a *pelda01.m* nevű fájlba mentettem. Ha a *pelda01* szót beírjuk a Command Window-ba, akkor lefut a szkriptünk. Ha van olyan sor a fájlban, amelyben egy változónak értéket adunk, pontosvessző nélkül a sor végén, akkor a változó az értékével együtt kiíródik a Command Window-ba.

A fenti programkódból jól látszik, hogy a *pelda01* beírása után az *u_c*, *u_d* és *p* változók értékei kiíródnak. A Command Window ekkor:

```
>> pelda01
u_c = -1.5000
u_d =  0.50000
p =  0.62466
>>
```

Az .m kiterjesztésű fájlunkat úgy is megírhatjuk, hogy a meghívásával egyidőben adunk értéket a bemenő változóknak. Eredményül a kimenő változó értékét kapjuk. A többi változót ekkor lokális változónak nevezzük.

| | |
|---|---|
| function NÉV<br>TÖRZS<br>endfunction | function NÉV (AL)<br>TÖRZS<br>endfunction |
| function VÉ = NÉV (AL)<br>TÖRZS<br>endfunction | function [VÉ] = NÉV (AL)<br>TÖRZS<br>endfunction |

*4.1. Táblázat - A függvény írásának formája különböző menniységű be-és kimeneti változó esetén*

- VÉ = egy visszatérési érték
- [VÉ] = több visszatérési érték
- AL = Argumentum lista

Az 1. Példa esetében egy visszatérési értékünk van (*p*), és négy bemenő adatunk *(a,sigma,c,d)*.

A módosított programkód a *pelda01f.m* fájlból: [Mellékletek: CD, *4.4*]

```
 1 %-- Function file: pelda01f (a,sigma,c,d)
 2 %
 3 %Valoszinusegszamitas: intervallum ertekek, szoras es varhato ertek
ismereteben.
 4 %Pelda adatok:
 5 %a = 80.200; varhato ertek
 6 %sigma = 0.04; szoras
 7 %c = 80.140;
 8 %d = 80.220; intervallum
 9
10 function p = pelda01f (a,sigma,c,d);
11 p = 0;
12 u_c = (c-a)/sigma;
13 u_d = (d-a)/sigma;
14 %u_c vizsgálata
15 if (u_c < 0) %2.63 képlet alapján átalakítás [Detrekői, 1991.]
16   fi_c = 1 - stdnormal_cdf(abs(u_c));
17 else
18   fi_c = stdnormal_cdf(u_c);
19 endif
20 %u_d vizsgálata
21 if (u_d < 0)
22   fi_d = 1 - stdnormal_cdf(abs(u_d));
23 else
24   fi_d = stdnormal_cdf(u_d);
25 endif
26 p = fi_d - fi_c;%Végeredmény
27 endfunction
```

Az első kommentezett sorok a *help pelda01f* parancs esetén kiíródnak a Command Window-ba, útmutatást nyújtva a felhasználónak a szkript tartalmáról, a függvény használatáról.

Az alábbi eredményt kapjuk a *help pelda01f* és *pelda01f(80.200,0.04,80.140,80.220)* parancsok kiadása után:

```
 1 >> help pelda01f
 2 'pelda01f' is a function from the file C:\Octave\geo\pelda01f.m
 3
 4 -- Function file: pelda01f (a,sigma,c,d)
 5
 6 Valoszinusegszamitas: intervallum ertekek, szoras es varhato ertek
ismereteben.
 7 Pelda adatok:
```

```
 8 a = 80.200; varhato ertek
 9 sigma = 0.04; szoras
10 c = 80.140;
11 d = 80.220; intervallum
12
13 Additional help for built-in functions and operators is
14 available in the online version of the manual.  Use the command
15 'doc <topic>' to search the manual index.
16
17 Help and information about Octave is also available on the WWW
18 at http://www.octave.org and via the help@octave.org
19 mailing list.
20 >> pelda01f(80.200,0.04,80.140,80.220)
21 ans = 0.62466
22 >>
```

A fenti feladatban megadott intervallum nem szimmetrikus, viszont a gyakorlati feladatok megoldásakor szimmetrikus intervallumot használunk leggyakrabban, melyet a $\sigma$ szórás $u$-szorosaként fejezünk ki. Elterjedt használni a szórás egy-, két- vagy háromszorosának megfelelő szélességű szimmetrikus intervallumot, melyek rendre *68,3%*, *95,4%* vagy *99,7%* valószínűséget jelentenek.

$$\begin{aligned} P=-\sigma\leqslant\xi-a\leqslant\sigma=0{,}6827 \\ P=-2\sigma\leqslant\xi-a\leqslant2\sigma=0{,}9545 \\ P=-3\sigma\leqslant\xi-a\leqslant3\sigma=0{,}9973 \end{aligned} \quad \text{(három-szigma szabály)} \tag{4.3}$$

Az előző feladat ellentettje, ha egy adott $p$ valószínűségi szinthez kell meghatároznunk az intervallumot. Akkor jutunk ebben az esetben egyértelmű megoldáshoz, ha feltételezzük, hogy a keresett intervallum a várható értékhez képest szimmetrikus és teljesül a

$$p=2\Phi(u)-1 \quad \text{egyenlet} \tag{4.4}$$

**2. Példa:** Az előző példában megadott értékek esetén keressük azt a várható értékhez képest szimmetrikus intervallumot, amelybe a valószínűségi változó $p$ = 0,5 valószínűséggel esik. Ekkor:

$a$ = 80,200 m, $\sigma$ = 0,04 m

A (4.4) képlet szerint Octave használatával egyszerű egyenletrendezéssel meghatároztam *Φ(u)* értékét, majd a beépített *stdnormal_inv* függvényt felhasználva megkaptam az $u$ értékét, mellyel kiszámíthatóak az intervallum határai a (4.5) képlet alapján.

$$c=a-u\sigma \qquad d=a+u\sigma \tag{4.5}$$

Ezt a feladatot is az előzőhöz hasonlóan oldottam meg, meghívható függvényként. A bemeneti érték jelen esetben *a, sigma, p* változók értékére módosul, és két visszatérési értékünk lesz, $c$ és $d$ intervallumhatárok.

A programkód a *pelda02f.m* fájlból: [Mellékletek: CD, *4.6*]

```
 1 %-- Function file: [c d] = pelda02f (a,sigma,p)
 2 %
 3 %Szimmetrikus intervallum kereses: varhato ertek, szoras es
valoszinuseg ismereteben.
 4 %Pelda adatok:
 5 %a = 80.200; varhato ertek
 6 %sigma = 0.04; szoras
 7 %p = 0.5
 8
 9 function [c,d] = pelda02f (a,sigma,p)
10 fi_u = (p+1)/2;
11 u = stdnormal_inv(fi_u); %inverz kell, mivel fi_u az adott.
12 %A végeredmény:
13 c = a - u*sigma;
14 d = a + u*sigma;
15 end
```

Eredmény a *[c d] = pelda02f (80.200,0.04,0.5)* parancs lefuttatása után:

```
1 >> [c d] = pelda02f (80.200,0.04,0.5)
2 c =  80.173
3 d =  80.227
4 >>
```

A megoldások jelen esetben a *c* és *d* változókba mentődnek. Természetesen más változóneveket is választhatunk. A 2. példa konkrét megoldását a *pelda02.m* fájl tartalmazza [Mellékletek: CD, *4.5*].

## 4.2. Két valószínűségi változó tapasztalati jellemzői

Két valószínűségi változó jellemzésére legtöbbször az egy valószínűségi változó jellemzőit használjuk fel, melyek:

1. A várható érték ($a$): $L_1, L_2, \ldots, L_k$ mintaelemek számtani közepe (mintaközép):

$$\bar{a} = \frac{1}{k}\sum_{i=1}^{k} L_i \tag{4.6}$$

2. Szórásnégyzet (variancia), melyet a tapasztalati szórásnégyzettel becslünk.

$$\bar{a} = \frac{1}{k-1}\sum_{i=1}^{k} \left(L_i - \bar{a}\right)^2 \tag{4.7}$$

3. Medián: az az érték, amelynél a valószínűségi változó 0,5 valószínűséggel vesz fel kisebb értéket.

4. Momentumok: a tapasztalati momentumokkal és a centrális tapasztalati momentumokkal becsüljük.

$$\bar{\alpha}_q = \frac{1}{k}\sum L_i^q \quad \textit{és} \quad \bar{\mu} = \frac{1}{k}\sum \left(L_i - \bar{a}\right)^q \tag{4.8}$$

$\bar{\alpha}_q$ = q-adik momentum és $\bar{\mu}$ = centrális momentum.

5. Ferdeségi együttható ($\gamma_1$) és lapultsági együttható($\gamma_2$):

$$\bar{\gamma}_1 = \bar{\mu}_3 / \bar{\mu}_2^{3/2} \quad és \quad \bar{\gamma}_2 = \bar{\mu}_4 / \bar{\mu}_2^2 - 3 \tag{4.9}$$

A változók kapcsolatát a következő tulajdonságokkal jellemezzük:

6. A kovariancia becslése a tapasztalati kovariancia:

$$\bar{c}_{12} = \frac{1}{k-1} \sum_{i=1}^{k} [(L_{1i} - \bar{a}_1)(L_{2i} - \bar{a}_2)] \tag{4.10}$$

7. A korrelációs együtthatót a tapasztalati korrelációs együtthatóval becsüljük:

$$\bar{r}_{12} = \frac{\bar{c}_{12}}{\bar{\sigma}_1 \bar{\sigma}_2} \tag{4.11}$$

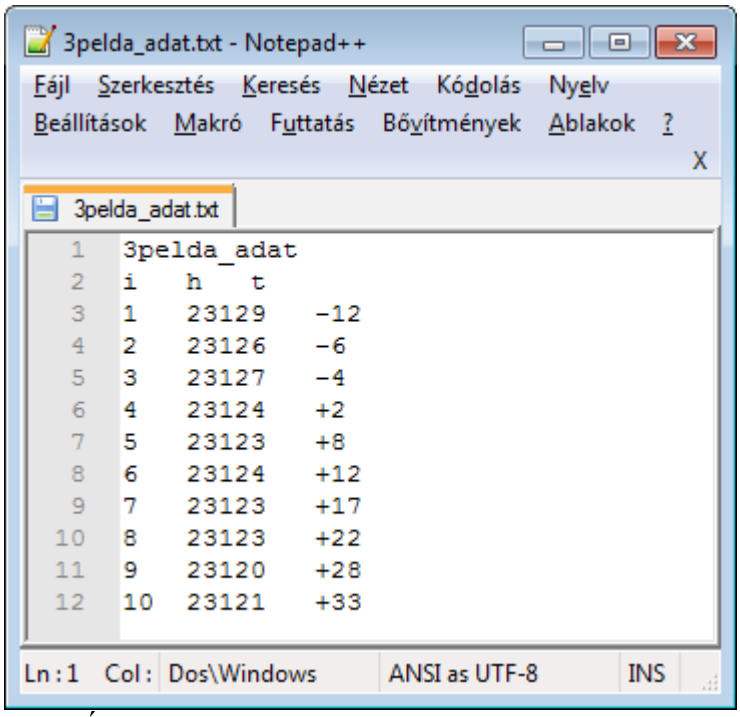
Két valószínűségi változó kapcsolatát szokás az M($\xi|\eta$) feltételes várható értékkel jellemezni, mely tapasztali megfelelője egy olyan függvény (egyenes), melyet az $\eta$ valószínűségi változó $\xi$-re vonatkozó regressziójának neveznek. Két valószínűségi változó esetén a regressziós egyenes egyenlete:

$$y = \bar{a}_2 + \bar{r}_{12} \frac{\bar{\sigma}_2}{\bar{\sigma}_1} (x - \bar{a}_1), \tag{4.12}$$

$$\text{ahol} \quad \bar{r}_{12} \frac{\bar{\sigma}_2}{\bar{\sigma}_1} = m \quad \text{az egyenes iránytangense} \tag{4.13}$$

**3. Példa:** Valamely épület hőmérséklet-változás hatására bekövetkezett mozgásának meghatározására mozgásvizsgálatot végeztek. A mozgásvizsgálatkor különböző $t_i$ hőmérsékleten meghatározták az épületen megjelölt pontok koordinátáit. Az egyik vizsgált pont esetében a meghatározott $h_i$ koordinátákat és a hozzájuk tartozó $t_i$ hőmérsékleteket az alábbi táblázat tartalmazza. A táblázat adatai alapján számítsuk ki a megfigyelt pont koordinátája és a mérési hőmérséklet kapcsolatát jellemző korrelációs együtthatót, majd írjuk fel a pont hőmérséklet-változás hatására bekövetkezett mozgását jellemző $h = b + m\ t$ regressziós egyenest.

Első lépésként a mért $h_i$ és $t_i$ adatokat egy szövegfájlba mentettem sorszámozással együtt az alábbi formátumban: [Mellékletek: CD, *2.1*]




*4.1. Ábra - A mért koordináták és hőmérséklet adatok [1.1]*

A fájl első két sora az úgynevezett header rész, mely információt ad a fájl további összetételéről, illetve tartalmáról.

Első sor a fájlnév, melyből látszik, hogy a szövegfájlban a 3-as feladat adatai találhatóak. Ezek az adatok az *i, h* és *t* részekre bonthatóak, melyek a *sorszám*, *koordináta* és *hőmérséklet* adatok, ebben a sorrendben. Az adatokat elválasztó jel a tabulátor. A megoldás során más elválasztójel is használható, mint például:

- ' ' vagy 'space'
- '\t' vagy 'tab'
- ',' vagy 'comma'
- ';' vagy 'semi'
- '|' vagy 'bar'

Octave programba beolvastam a szövegfájlban tárolt adatokat az *fscanf* függvény használatával, megadva a megfelelő paramétereket. Az elérési útvonal meghatározása során hasznos a *filesep* parancs használata, mely megadja az aktuális platformon használt elválasztó karaktert ('/' vagy '\'). Az első két sort külön kezeltem, stringként, majd az *i,h,t* oszlopokat fixpontos számként olvastam be. A beolvasás után egy 3x10 méretű mátrixot

kapunk, mely transzponálással egy lépésben 3 különálló részre bontható, melyek *i,h* és *t* változókba mentettem. A beolvasás az első elemtől végtelenig tart. A gyakorlatban a végtelenig tartó olvasás az utolsó sort jelenti. Előnynek számít így, hogy utólag is tölthetünk fel adatot, frissíthetjük a szövegfájlt, az eredmények ennek függvényében változnak.

Először a koordináták és hőmérsékletek mintaközepét számoltam ki a (4.6) képlet szerint. A tapasztalati szórásokat a (4.7) képlet pozitív gyökeként kaptam meg, ezek a programkódban *sigma_1* (koordináta) és *sigma_2* (hőmérséklet) változók. A kovariancia érték (4.10) képlet szerinti meghatározása után kiszámoltam a korrelációs együttható értékét, melyből a regressziós együttható meghatározható. Az adatok felhasználásával a (4.12) képlet szerint meghatároztam a regressziós egyenes két tetszőleges pontjának *h* értékét a minimum-3°C és maximum+3°C hőmérsékletű pontokban. A két pontot felhasználva ábrázoltam a *plot* függvény használatával a regressziós egyenest ezen az intervallumon, illetve a 10 adott hőmérséklet-koordináta párost.

A számított adatok:

$\bar{a}_1 = 23124$ mm $\quad \bar{a}_2 = 10$ °C

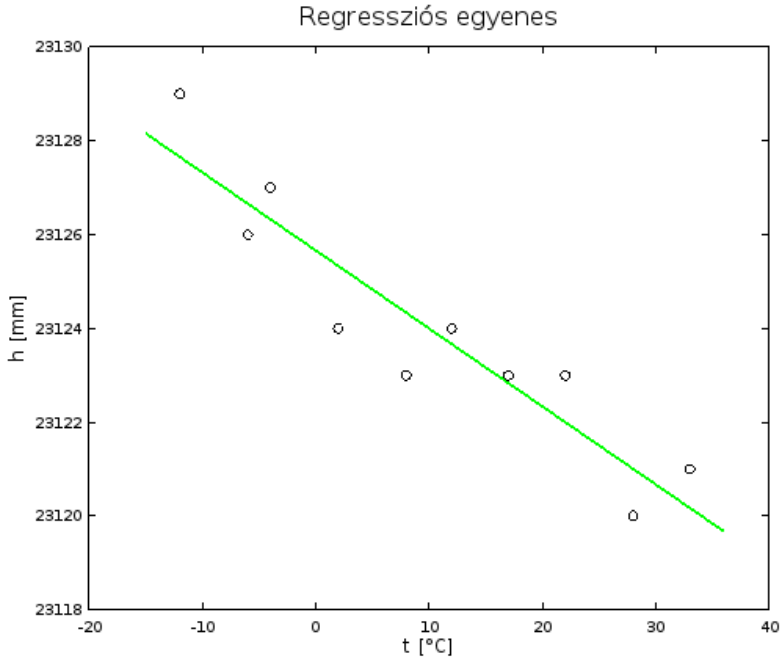
$\bar{\sigma}_1 = 2{,}7080$ mm $\quad \bar{\sigma}_2 = 15{,}107$ °C

$\bar{c}_{12} = -38$ mm°C $\quad \bar{r}_{12} = -0{,}92887$

$m = -0{,}1665$

$p_1 = 36$ °C $\quad p_2 = -15$ °C

$h_1 = 23120$ mm $\quad h_2 = 23128$ mm (regressziós egyenes két pontja)




*4.2. Ábra - Regressziós egyenes [1.3]*

A programkód: [Mellékletek: CD, *4.7*]

```
%Adatok beolvasása, szétválasztása.
filename = '3pelda_adat.txt';%Fájlnév, mely az adatokat tartalmazza.
DIR_PATH = 'C:\Octave\geo';
file_path=[DIR_PATH filesep filename];%Elérési útvonal a fájlhoz.
fid=fopen(file_path);%Fájl megnyitása olvasásra.
fname=fscanf(fid,'%s',1);%Header fájl első sorának beolvasása (string).
dataname=fscanf(fid,'%s',3);
data=fscanf(fid,'%f %f %f',[3 inf]);
fclose(fid);
i = data(1,:)';%sorszámozás
h = data(2,:)';%koordináta adatok
t = data(3,:)';%hőmérséklet adatok
%Számítások:
a1=1/size(i,1)*sum(h);%h_i koordináták mintaközepe.
a2=1/size(i,1)*sum(t);%t_i hőmérséklet adatok mintaközepe.
%sigma1
f1 = 0;
sum_h = 0;
for f1 = 1:size(i,1);
s_f1 = (h(f1) - a1)^2;
sum_h = sum_h + s_f1;
endfor
sigma_1 = abs(sqrt( 1/(size(i,1)-1)*sum_h));%koord. tapasztalati szórás
%sigma_2
f2 = 0;
sum_t = 0;
for f2 = 1:size(i,1);
s_f2 = (t(f2) - a2)^2;
sum_t = sum_t + s_f2;
endfor
sigma_2 = abs(sqrt( 1/(size(i,1)-1)*sum_t));%hőmérséklet tap. szórása
%kovariancia számítása (c)
fc = 0;
sum_c = 0;
for fc = 1:size(i,1);
s_fc = (h(fc)-a1)*(t(fc)-a2);
sum_c = sum_c +s_fc;
endfor
c = 1/(size(i,1)-1)*sum_c;
%korrelációs együttható számolása (r):
r = c / (sigma_1*sigma_2);
%regressziós együttható (m):
m = r*sigma_1/sigma_2;
%regressziós egyenes == y=a1+m(x-a2):
%h = a1+m*(t-a2);
p1 = t(end)+3;
p2 = t(1)-3;
h1 = a1 + m*(p1-a2);
h2 = a1 + m*(p2-a2);
%pontok és az egyenes kirajzolása
figure(3);
clf;
hold on;
plot([p1 p2],[h1 h2],'linewidth',2,'g');
plot(t,h,'ko');
plot([0 0],[23118 23130]);
title('Regressziós egyenes','fontsize',16);
xlabel('t [°C]','fontsize',12);
ylabel('h [mm]','fontsize',12);
```

A feladatot meghívható függvényként is elkészítettem, *pelda03f.m* néven [Mellékletek: CD, *4.8*]. Ekkor elég megadni a megfelelően formázott fájl elérési útvonalát és a fájlnevet, például *pelda03f ('C:\Octave\geo','3pelda_adat.txt')*. Ezután a regressziós egyenest kajuk eredményül egy ábrán kirajzolva.

## 4.3. Kiegyenlítés közvetítő egyenletekkel (II. Kiegyenlítési csoport)

A közvetítő egyenleteket akkor használjuk, ha a keresett paramétereket más mennyiségekre végzett mérések eredményei alapján szeretnénk becsülni.

A feladat megoldása során a $\xi_1, \xi_2, ..., \xi_n$ jelöli a valószínűségi változókat, a mérési eredményeink pedig $L_1, L_2, ...,L_n$, a kiegyenlített mérési eredménynek pedig $U_1, U_2, ..., U_n$.

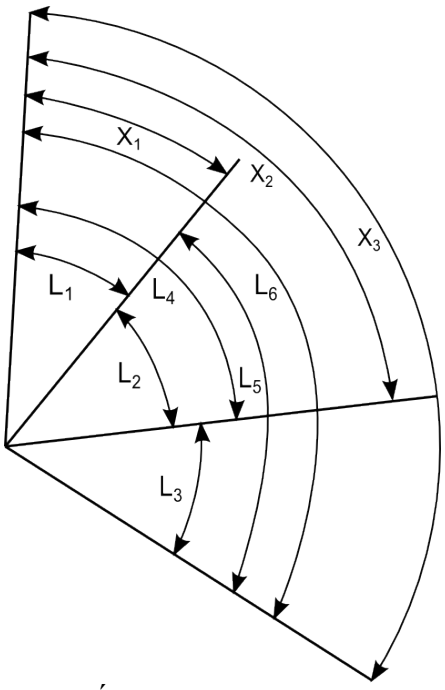
$$U_i = L_i + v_i \tag{4.14}$$

A (4.14) egyenletből látszik, hogy a $v_1, v_2, ..., v_n$ javítások segítségével meghatározhatóak a kiegyenlített mérési eredmények.

Az $\eta_1, \eta_2, ..., \eta_n$ a meghatározandó paraméterek, melyek kiegyenlítésből nyert értékei $X_1, X_2, ..., X_n$. Általában az elvégzendő feladatoknál az $x_1, x_2, ..., x_n$ változásokat határozzuk meg a paraméterek előzetes értékeiből. Ezek közötti összefüggés:

$$X_j = X_{j0} + x_j \tag{4.15}$$

**4. Példa:** Valamely műszerállásponból kiinduló négy irány között megmértük a 4.3. ábrán feltüntetett $L_1, ..., L_6$ szögértékeket. Az $L_1, L_2, L_3$ szögek méréséhez $\mu=2''$ középhibával jellemezhető, az $L_4, L_5, L_6$ szögek méréséhez pedig $\mu=4''$ középhibával jellemezhető műszerfelszerelést használtunk. Az egyes mérések függetlennek tekinthetők. A mérési eredmények alapján határozzuk meg a 4.3. ábrán feltüntetett $X_1, X_2, X_3$ paraméternek tekintett szögeket, továbbá a kiegyenlített mérési eredményeket!




*4.3. Ábra - Mért és számított szögértékek [1.4]*

A mérési eredmények a következők:

| | |
|---|---|
| $L_1 = 30°40'24''$ | $L_4 = 72°46'40''$ |
| $L_2 = 42°06'12''$ | $L_5 = 88°18'27''$ |
| $L_3 = 46°12'20''$ | $L_6 = 118°58'59''$ |

A számítás általános lépései:

1. Az előzetes értékek felvétele, jelen esetben:

$X_{10} = 30°40'20''$

$X_{20} = 72°46'40''$

$X_{30} = 118°58'50''$

2. A közvetítő egyenletek felírása, melyek lineárisak:

$$\underset{(n,1)}{L} + \underset{(n,1)}{v} = \underset{(n,r)}{A} * \underset{(r,1)}{x} + \underset{(n,1)}{a_0} \tag{4.16}$$

| | |
|---|---|
| $U_1 = X_1$ | $U_4 = X_2$ |
| $U_2 = X_2 - X_1$ | $U_5 = X_3 - X_1$ |
| $U_3 = X_3 - X_2$ | $U_6 = X_1$ |

3. A javítási egyenletek:

$$\begin{aligned}
v_1 &= x_1 + (X_{10} - L_1) = x_1 - 4 \\
v_2 &= -x_1 + x_2 + (X_{20} - X_{10} - L_2) = -x_1 + x_2 + 8 \\
v_3 &= -x_2 + x_3 + (X_{30} - X_{20} - L_3) = -x_2 + x_3 - 10 \\
v_4 &= x_2 + (X_{20} - L_4) = x_2 \\
v_5 &= -x_1 + x_3 + (X_{30} - X_{10} - L_5) = -x_1 + x_3 + 3 \\
v_6 &= x_3 + (X_{30} - L_6) = x_3 - 9
\end{aligned} \tag{4.17}$$

A javítási egyenleteket mátrixos alakban is fel tudjuk írni:

$$\begin{gathered}
\underset{(n,1)}{v} = \underset{(n,r)}{A} * \underset{(r,1)}{x} + \underset{(n,1)}{l} \quad , ahol \\
\underset{(n,1)}{l} = \underset{(n,1)}{L} - \underset{(n,1)}{a_o}
\end{gathered} \tag{4.18}$$

4. Mérési eredmények súlymátrixának felvétele:

A méréseinket ebben a feladatban különböző pontosságúnak és függetlennek tekintjük, a súlymátrix diagonálmátrix lesz. A súlyok a (4.19) összefüggésből határozhatóak meg.

$$p_i = \frac{c^2}{\mu^2} \tag{4.19}$$

A súlymátrix felvételekor még további két eset fordulhat elő.

a) A mérések azonos pontosságúak és korrelálatlannak tekinthetők, a súlymátrix ekkor egységmátrix lesz.

b) A mérések korreláltak, ekkor a súlymátrix $\mathbf{C}^{-1}$-el egyező, ahol **C** a mérések a priori kovarianciamátrixa.

5. A normálegyenlet felállítása, a súlymátrix típusának figyelembe vételével:

$$\underset{(3,3)}{(A' * P_{LL} * A)} * \underset{3,1}{x} - \underset{(3,1)}{(A' * P_{LL} * l)} = \underset{(3,1)}{0} \tag{4.20}$$

6. A normálegyenlet megoldásából az x változás meghatározása:

Ha az **A** mátrix és az **N** mátrix ($\mathbf{A'*P_{LL}*A}$ jelen esetben) rangja megegyezik a paraméterek számával, akkor az $\mathbf{N}^{-1}$ mátrix létezik. Ekkor:

$$\underset{(r,1)}{x} = \underset{(r,n)}{N^{-1}} * \underset{(n,1)}{n} = \underset{(r,r)}{(A' * P_{LL} * A)^{-1}} * \underset{(r,1)}{(A' * P_{LL} * l)} \tag{4.21}$$

Ha az **A** mátrix és az **N** mátrix rangja kisebb az ismeretlen paraméterek számánál, akkor az $\mathbf{N}^{-1}$ mátrixot nem tudjuk értelmezni. Ekkor **N** egy általánosított inverze, az $\mathbf{N}^{+}$ pszeudoinverz felhasználásával jutunk megoldáshoz. Ekkor:

$$\underset{(r,1)}{x} = \underset{(r,r)}{N^{+}} * \underset{(r,1)}{n} = \underset{(r,r)}{(A' * P_{LL} * A)^{+}} * \underset{(r,1)}{(A * P_{LL} * l)} \tag{4.22}$$

7. A paraméterek kiegyenlített értékének meghatározása:

$$\underset{(r,1)}{X} = \underset{(r,1)}{X_0} + \underset{(r,1)}{x} \tag{4.23}$$

8. A mérési javítások kiszámítása:

$$\underset{(n,1)}{v} = \underset{(n,r)}{A} * \underset{(r,1)}{x} - \underset{(n,1)}{l} \tag{4.24}$$

9. A kiegyenlített mérési eredmények meghatározása:

$$\underset{(n,1)}{U} = \underset{(n,1)}{L} + \underset{(n,1)}{v} \tag{4.25}$$

10. Ellenőrzés:

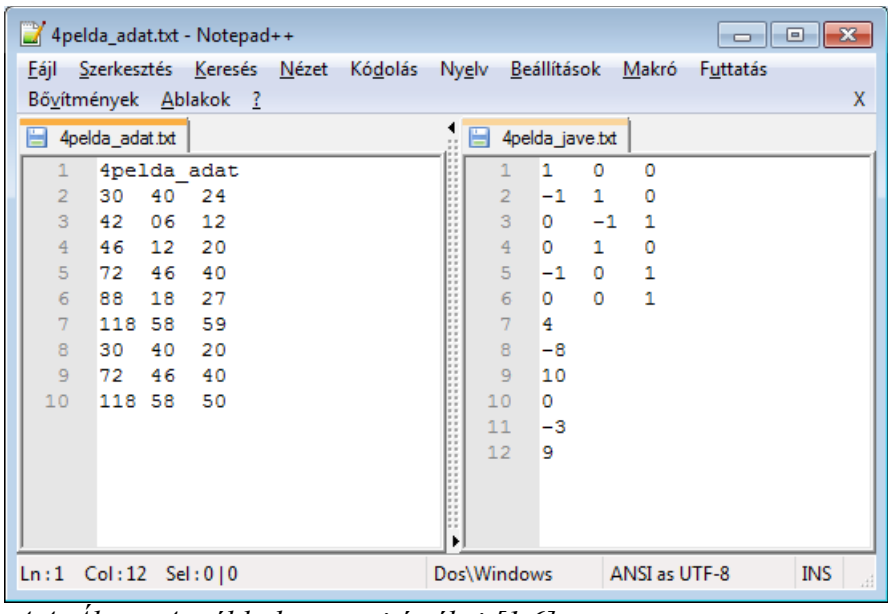
Az ellenőrzés kétféleképp történik.

a) A paramétereknek és a kiegyenlített mérési eredményeknek behelyettesítésével az eredeti közvetítő egyenletekbe.

b) A javítások és a változások között fennálló összefüggés alapján:

$$\underset{(1,n)}{v'} * \underset{(n,n)}{P_{LL}} * \underset{(n,1)}{v} = - \underset{(1,n)}{l'} * \underset{(n,n)}{P_{LL}} * \underset{(n,r)}{A} * \underset{(r,1)}{x} + \underset{(1,n)}{l'} * \underset{(n,n)}{P_{LL}} * \underset{(n,1)}{l} \tag{4.26}$$

A feladat Octave-ban való megoldásához először a mérési adatokat és a javítási egyenletben szereplő **A** mátrixot és **l** oszlopvektort elmentettem két szövegfájlba [Mellékletek: CD, *2.2, 2.3*]. A szövegfájlok felépítése hasonlóan egyszerű a 3. példában szereplő megoldáshoz.




*4.4. Ábra - 4. példa bemeneti értékei [1.6]*

Az adatokat egy *adat* nevű változóba olvastam be. Mivel a későbbi számolások során külön kell felhasználni az $L_i$ és $X_{i0}$ értékeket, ezért ezeket szétválasztottam.

$$L = adat([1\ 2\ 3\ 4\ 5\ 6], :)$$
$$X_0 = adat([7\ 8\ 9], :) \tag{4.27}$$

A súlyok kiszámításához $c^2$ értéket az *lcm* (least common multiple – legkisebb közös többszörös) függvény használtával állapítottam meg, majd a beépített *diag* függvénnyel létrehoztam a *P_LL* diagonális súlymátrixot. A normálegyenlet felírása után kiszámoltam az *x* változásokat. Az eredményt a "\" művelet használatával kaptam meg.

A keresett paraméterek meghatározásához készítenem kellett két függvényt, melyek feladata fok-perc-másodperc érték átváltása tizedfokba, illetve fordított esetben a tizedfok megadása fok-perc-másodperc értékben. Ezen függvényeket *atvalt_fok.m* és *atvalt_fpm.m* fájlokba mentettem egy rövid leírással együtt [Mellékletek: CD, *4.1, 4.2*]

*atvalt_fok.m* fájl tartalma:

```
%-- Function file: atvalt_fok(fok_be)
%
%Az atvalt fv. a bemeno fok perc masodperc erteket
%valtja at tizedfokba.
%Bemeno adat:
%   vektorkent
%   matrixkent,ahol 1. oszlop = fok,
% 2. oszlop = perc, 3. oszlop = masodperc
function fok_ki = atvalt_fok(fok_be)

fok_ki = fok_be(:,1) + fok_be(:,2)/60 + fok_be(:,3)/3600;

end
```

*atvalt_fpm.m* fájl tartalma:

```
%-- Function file: atvalt_fpm(fok_be)
%
% Az atvalt_fpm fuggveny tizedfokbol
% fok perc masodperc ertekekbe szamol at.
% A bemeno adat: skalar
%                vektor
% A kimenet egy vektor (skalar eseten)
% vagy egy matrix (vektor eseten).
function fpm_ki = atvalt_fpm(fok_be);

f = fix(fok_be);
p = fix((fok_be-f) * 60);
m = ((fok_be-f)*60-p)*60;
fpm_ki = [f p m];

end
```

Ezek felhasználásával előbb tizedfokba váltottam a szögértékeket, melyek mátrixban voltak tárolva. Az átalakítás hatására a 3 oszlopos mátrix helyett egy oszlopvektort kaptunk eredményül, így ehhez már hozzátudjuk adni a változások 3600-ad részét. Az összeadás után megkapjuk a keresett paramétereket tizedfokban.

A javításokat a (4.24) képlet szerint számoltam, majd ennek felhasználásával a (4.25)

képlet alapján megkaptam a kiegyenlített mérési eredményeket.

A feladatot meghívható függvényként is elkészítettem, ahol megadható az elérési útvonal, illetve a két szövegfájl neve. Így hasonló feladattípus más mérési eredményekkel szintén elvégezhető [Mellékletek: CD, *4.10*].

A programkód: [Mellékletek: CD, *4.9*]

```
 1 %A mérési eredmények betöltése
 2 DIR = '.';
 3 filename = '4pelda_adat.txt';
 4 filename2 = '4pelda_jave.txt';
 5 file_path1 = [DIR filesep filename];%Elérési útvonal a fájlhoz.(Mérési
adatok)
 6 file_path2 = [DIR filesep filename2];%Elérési útvonal a fájlhoz.
(Javítási egyenlet)
 7 fid1 = fopen(file_path1);%Fájl megnyitása olvasásra.
 8 fid2 = fopen(file_path2);%Fájl megnyitása olvasásra.
 9 fname = fscanf(fid1,'%s',1);%Header fájl első sorának beolvasása
stringként.
10 adat = fscanf(fid1,'%f %f %f',[3 Inf])';%Adatok beolvasása
11 A = fscanf(fid2,'%f %f %f',[3 6])';%Javítási egyenlet, A mátrix
12 l = fscanf(fid2,'%f');%Javítási egyenlet, l mátrix
13 fclose(fid1);
14 fclose(fid2);%szövegfájlok bezárása
15
16 A_t = A';
17 L = adat([1 2 3 4 5 6],:);
18 X0 = adat([7 8 9],:);%L és X0 mérések szétválasztása
19 %A súlyok kiszámítása:
20 u1 = 2;
21 u2 = 4;
22 c2 = lcm(u1^2,u2^2);
23 p1 = p2 = p3 = c2/u1^2;
24 p4 = p5 = p6 = c2/u2^2;
25 p = [p1 p2 p3 p4 p5 p6];
26 P_LL = diag(p);
27 %Normálegyenlet felállítása:
28 A_t_P_LL = A_t*P_LL;
29 A_t_P_LLA = A_t_P_LL*A;
30 A_t_P_LLl =(-1).*(A_t_P_LL*l);
31 %Az egyenletrendszer megoldása, a változások:
32 x = -(A_t_P_LLA\A_t_P_LLl);
33 %Az előzetes értékek és a változások alapján a keresett paraméterek:
34 X0_fok=atvalt_fok(X0);
35 X_fok = X0_fok + x/3600;
36 X = atvalt_fpm(X_fok);
37 %A javítások:
38 v = A*x-l;
39 %A kiegyenlített mérési eredmények:
40 L_fok = atvalt_fok(L);
41 U_fok = L_fok + v./3600;
42 U = atvalt_fpm(U_fok)
43 %Ellenőrzés, javítások és változások közötti összefüggés:
44 ell_1 = (v'*P_LL*v);
45 ell_2 = -(l'*P_LL*A*x)+l'*P_LL*l;
46 ell = ell_1 - ell_2
```

A kapott eredmény:

```
 1 >> pelda04
 2 U =
 3
 4     30.0000    40.0000    25.2293
 5     42.0000     6.0000    12.0976
 6     46.0000    12.0000    19.4293
 7     72.0000    46.0000    37.3268
 8     88.0000    18.0000    31.5268
 9    118.0000    58.0000    56.7561
10
11 ell =   4.9738e-014
```

Az ellenőrzést a (4.26) képlet alapján végeztem a szkriptben. A képlet bal és jobb oldala közötti eltérés az *ell* változó értéke, mely $4.9738*10^{-14}$, így az eltérés 0-nak tekinthető, a (4.26) egyenlet teljesül.

## 4.4. A GNU Octave további geodéziai felhasználási lehetőségei

További Octave funkciók nyújthatnak megoldást komplexebb geodéziai feladatok elvégzésére.

Ilyen feladat lehet mérőállomással mért mérési állomány beolvasása, a mérés feldolgozása, mérési eredmények, koordináták szétválasztása vagy akár a pontok transzformálása, grafikus megjelenítése. A megoldáshoz persze szükséges a mérési állomány összetételének ismerete.

A kétdimenziós adatok mellett háromdimenziós adatokkal is dolgozhatunk. Ilyen feladat lehet például GPS adatok vagy lézerszkennerrel mért pontfelhők beolvasása, grafikus megjelenítése.

Octave-ban lehetőségünk van képek beolvasására, megjelenítésére is. A képek felhasználásával létrehozhatunk animációkat különböző időintervallumra eső időjárási adatokkal kombinálva.

# Összegzés

Szakdolgozatom célja a szoftverlicencek ismertetése, a mérnöki alkalmazási területhez kapcsolódó szabadszoftverek, ezen belül is a GNU Octave bemutatása volt.

Jól látszik, hogy a szabadszoftverek elterjedését nagyban segíti a gyors fejlődés, funkcionális bővítés, nagyobb stabilitás és kompatibilitás biztosítása több rendszeren, mindezt ingyenesen.

A mérnöki feladatok elvégzésére főként tulajdonosi szoftverek terjedtek el, viszont a fent leírtak tükrében elmondható, hogy a szabadszoftverek jó, olcsó alternatívát jelentenek a tulajdonosi szoftverek kiváltására. Ez főként elmondható egy olyan programozási nyelvről, mint az Octave, mivel itt akár mi magunk is írhatunk, definiálhatunk függvényeket az Octave saját programozási nyelvén, így megoldást nyújtva különböző mérnöki feladatokra.

A megvalósított alkalmazási példák bizonyítják, hogy a szoftver precíz, komplex mérnöki alkalmazási területeken megfelelően hsználható. Könnyen és gyorsan megoldható vele valószínűségszámítási feladat a beépített függvények használatával, illetve mérési fájlok olvasása és feldolgozása, az adatok ábrázolása is egyszerűen elvégezhető.

Fontosnak tartom a mérnöki számítások szempontjából a szabadszoftveres támogatást, legyen szó akár kész szoftverről vagy matematikai programozási nyelvről, mint például az Octave-ról. Kifejezetten előnyösnek tartom, hogy saját magunknak (vagy csoportosan) kell írni, előállítani egy feladathoz szükséges megoldást, mivel így nem kell alkalmazkodnunk a kész szoftverek adta korlátokhoz, illetve saját igényeinknek megfelelő szoftvereket, rendszereket alkothatunk.

# Irodalomjegyzék


- Bácsatyai L. 2010: Kiegyenlítő számítások,NyME-Geo, Székesfehérvár
- Detrekői Á. 1991: Kiegyenlítő számítások, Tankönyvkiadó, Budapest
- J. W. Eaton, D. Bateman, S. Hauberg 2011: Edition 3 for Octave version 3.8.0, Network Theory Ltd., USA
- S. Gisbert 2013: eKönyv – Matlab, Typotex Elektronikus Kiadó Kft.
- B. Stroustrup 2001: A C++ Programozási nyelv I-II., Kiskapu kiadó


## Internetes források


- http://www.gnu.org/philosophy/license-list.html
- http://www.gnu.org/software/octave/
- http://en.wikipedia.org/wiki/Free_software_licence
- http://nyelvek.inf.elte.hu/leirasok/Octave/
- http://szabadszoftver.kormany.hu/
- http://fsf.hu/
- http://www.gnu.hu/
- http://www.lipsz.hu/

# Mellékletek

## Digitális mellékletek

1. Ábrák
    1.1. 3pelda_adat.png
    1.2. 3pelda_regegyenes.png
    1.3. 3pelda_regegyenes_crop.png
    1.4. 4pelda_abra.png
    1.5. 4pelda_abra.svg
    1.6. 4pelda_txt.png
    1.7. Octave_command_history.png
    1.8. Octave_command_window.png
    1.9. Octave_editor.png
    1.10. Octave_file_browser.png
    1.11. Octave_logo.png
    1.12. Octave_print_screen.png
    1.13. Octave_workspace.png
    1.14. szabad-cd.png
2. Adatok
    2.1. 3pelda_adat.txt
    2.2. 4pelda_adat.txt
    2.3. 4pelda_jave.txt
3. GNU Octave 3.8.1
    3.1. octave.pdf
    3.2. octave-3.8.1-1-installer.exe
4. M-fájlok
    4.1. atvalt_fok.m
    4.2. atvalt_fpm.m
    4.3. pelda01.m
    4.4. pelda01f.m

4.5. pelda02.m

4.6. pelda02f.m

4.7. pelda03.m

4.8. pelda03f.m

4.9. pelda04.m

4.10. pelda04f.m

5. Szakdolgozat

5.1. Szakdolgozat.odt

5.2. Szakdolgozat.pdf