NTI_Wavelet_Tools/AUG_DATA programcsomag dokumentáció

Papp Gergely
papp@reak.bme.hu

BME NTI

2010. augusztus 17.

Tartalomjegyzék

1. fejezet

Bevezetés

A következő pontok kerülnek majd kifejtésre itt:
A programrendszer célja
Telepítés (Teljes mappastruktúra és/vagy telepítőscript)
Adatfeldolgozó környezet
Jelölésrendszer
Jelmagyarázat
Rövidítések jegyzéke
IPP informatikau dokumentáció: http://www.rzg.mpg.de
AUG belső oldalak: https://www.aug.ipp.mpg.de/wwwaug/

1.1. A programrendszer elérhetősége

A programrendszer az NTI_Wavelet_Tools részét képezi.

1.1.1. Közvetlenül a szerzőktől

NWT: pokol@reak.bme.hu
AUG: papp@reak.bme.hu, magyarkuti@reak.bme.hu

1.1.2. SVN

A rendszer SVN repository-ját a `deep.reak.bme.hu` számítógép tárolja:
`svn+ssh://deep.reak.bme.hu//svn/NTI_Wavelet_Tools`
Belépés csak `SSH`-n keresztül lehetséges, ezért hozzáférést a szerzőktől kérhetünk.

1.1.3. Az IPP AFS-en

/afs/ipp-garching.mpg.de/home/p/pog/idl/NTI_Wavelet_Tools

1.2. Adatlekérő környezet

1.2.1. Távoli futtatás

Az adatlekérést a Max Planck Institut für Plasmaphysik (röv: IPP) rendszerébe belépve kell megtenni. Az adatlekérő programok csak az IPP garchingi campusának valamelyik gépén futtatva fognak működni. A belépéshez szükséges jogosultság megszerzésének módja nem része ezen dokumentációnak, általában az adott téma vezetőjétől lehet kérni. Ha rendelkezünk account-al, akkor belépni távolról `SSH` (Secure SHell) kapcsolaton keresztül tudunk. Windowsra az erre általánosan használt program a `Putty` (http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html, itt a `putty.exe` fájlt töltsük le). SSH kapcsolatot unixon a konzolból kiadott `ssh usernev@gepnev` parancs segítségével hozhatunk létre.

A belépéshez szükséges egy kapugép, amire csatlakozhatunk. Ez egy olyan számítógép, amely az IPP campuson kívülről is elérhető SSH kapcsolaton. Ilyen kapugép pl a `gate1.aug.ipp.mpg.de`. A belépéshez még szükséges a már említett felhasználónév-jelszó páros. A `Putty` használata esetén a program megnyitása után gépeljük/másoljuk be a kapugép nevét az ablakba, majd nyomjunk `enter`-t. Ekkor létrejön a kapcsolat a távoli géppel, amely azonosítást kér. Egy felugró ablak kérdezi meg felhasználónevünket és jelszavunkat. Adjuk meg, és már bent is kell legyünk a rendszerben, amire egy gépspecifikus üdvözlőszöveg emlékeztet minket.

Az IPP-ben főleg SUN gépek üzemelnek Solaris 9 vagy 10 operációs rendszerrel. A Solaris unix alapú rendszer, ezért az alap fájlműveleti parancsok ugyanazok mint bármelyik unixon, pl. linuxon. Ezeket interneten gyorsan megtalálhatjuk a google-be "linux alapparancsok" kifejezésre keresve. Két példa: http://www.doksi.atw.hu/ és http://www.hogyan.org/linux-alapparancsok. Ezek segítségével már képesek leszünk alap parancsokat kiadni és munkánkat elkezdeni. Egy tipp: a linuxokon általában a `bash` parancsértelmező a default, még a SUN gépeken nem. A linux alapokat tartalmazó leírások feltételezik a `bash` használatát, ezért induláskor praktikus lehet nekünk is ezt használni a SUN-okon default `ksh` vagy `tcsh` helyett. A bash parancsértelmezőre jellemző, hogy egy $ jel áll a parancssor elején. Amennyiben a SUN gépeken is ezt a parancsértelmezőt kívánjuk használni, csak adjuk ki a `bash` parancsot. Egy példa a belépésre unix rendszerről:

```
gm:~ geri$ ssh pog@gate1.aug.ipp.mpg.de
Last login: Tue Jul 21 10:49:18 2009 from labor1.reak.bme
Sun Microsystems Inc.   SunOS 5.10      Generic January 2005
AFS-Client 9.Feb.2009  -  SunStudio12
Sun Microsystems Inc.   SunOS 5.10      Generic January 2005
AFS-Client 9.Feb.2009  -  SunStudio12
tcsh: using dumb terminal settings.
pog@seaug2 1) bash
bash-3.00$
```

Láthatjuk az `ssh` parancs kiadását az adott gépre. Ennél a felállásnál a rendszer nem kért jelszót, mert SSH kulcsos azonosítást használtam. (Az ssh kulcsos azonosításnál nem kell jelszót megadni, de **használata csak haladó felhasználóknak javasolt!** Bővebben lásd http://sial.org/howto/openssh/publickey-auth/). Ezután üdvözlőszöveget látunk, majd egy sor figyelmeztet, hogy `tcsh` parancsértelmezőt használunk. Erre a sor végi `')'` is emlékeztet. Ezután a `bash` paranccsal hívtam életre a `bash` parancsértelmezőt, amire a sorvégi $ jel emlékeztet.

1.2.2. A fájlrendszer elérése – klog

Amennyiben kívülről belépünk egy kapugépre, akkor parancsokat már adhatunk ki, de a központi fájlrendszert még nem érjük el. Ez a kapcsolat biztonsági okokból nem jön létre automatikusan ha az IPP campuson kívülről lépünk be. Ez azért van így, mert lehetséges ssh-val a jelszó megadása nélkül is belépni, ami biztonsági problémákat okozhat ha pl. egy így beállított gép rossz kezekbe kerül. Ezért távolról belépve még külön szükséges hogy az IPP-ben alkalmazott Andrew elosztott fájlrendszer (`afs`) hozzáférésünket is létrehozzuk, addig nem tudjuk írni a fájlokat és sok fájlt olvasni sem tudunk. A fájlrendszer eléréséhez szükségünk van egy úgynevezett `afs token`-re. Ennek megszerzéséhez ki kell adni a terminalban a `klog` parancsot, majd megadni az account jelszavunkat. Ekkor az adott kapugép csatlakozik az elosztott fájlrendszert kiszolgáló szerverekhez, és ezután már elérjük adatainkat. Fontos tudni, hogy az `afs` rendszeren bármelyik gépről is dolgozunk, a saját fájljainkat mindenhonnan ugyanúgy érjük el. Ez az elosztott fájlrendszer egyik előnye.

1.2.3. Fájlmozgatás - AFS kliensszoftver, SCP

A fájlok fel- illetve lemásolása történhez valamilyen scp (Secure CoPy) program segítségével. Az scp ssh kapcsolaton keresztül másol fájlokat. Unixon

erre alkalmas az `scp` parancs, windowson praktikus pl. a `winscp` használata. A `winscp`-t a http://winscp.net/ oldalról tölthetjük le, telepítési és használati útmutatókat is találunk ezen a honlapon hozzá. A `winscp`-vel annyi probléma lehet, hogy a `klog` parancs kiadására nem vagyunk képesek vele, ezért előfordulhat hogy problémát észlelünk fájlok fel- vagy lemásolása közben.

Második megoldás, hogy telepítünk egy afs kliensszoftvert a gépünkre (http://www.openafs.org), amivel hálózati meghajtóként tudjuk csatlakoztatni az ottani fájlrendszert. Ez akkor hasznos, hogyha gyakran másolunk adatokat fel– és le. A kliensszftver telepítése után a cella szervernek az `ipp-garching.mpg.de` szervert állítsuk be. A hálózati meghajtó beállításához kell az elérési út. Az `afs` úgy működik, hogy a kliensünkkel a világ bármelyik `afs` rendszerére csatlakozhatunk, amennyiben oda van érvényes `token`-ünk. Ezt a tokent szereztük meg a `klog` paranccsal a távoli gépre belépve (1.2.2). A kliensnél `token`-t a "Get token", vagy hasonló paranccsal szerezhetünk. Ezután a fájljainkat a `/afs/ipp-garching.mpg.de/home/p/pog` útvonalon érjük el, ezt kell megadni a hálózati meghajtó beállításakor az `afs` kliensszoftvernek, mint elérési út. Itt a példában a `pog` accountot használtuk. Más account használata esetén a `/p/` helyére az adott account első betűjét, a `pog` helyére az adott account nevét tegyük. Pl. ha az account neve `asd` lenne, úgy az elérési utat `/afs/ipp-garching.mpg.de/home/a/asd`-re kellene változtatnunk.

Ha unix alapú rendszeren dolgozunk, úgy a kliens telepítésekor létrejön egy `/afs` mappa így a fenti elérési út egyből érvényes lesz a gyökérből.

Harmadik megoldás, hogy az IPP rendszeréről közvetlenül küldjük az adatot az `scp` parancs segítségével egy olyan gépre, ami ezt fogadni tudja (pl egy unix szerver). Például az `gp_sxr_savedata` programban (lásd 2.1) ez az opció be van építve. Ekkor a következő parancs kerül kiadásra a távoli oldalon:

```
scp eleresi_ut/fajl.neve user@gep:/teljes_eleresi_ut
```

Lássunk erre egy példát:

```
pog@seaug2 6) scp ~/mappa/valami.txt user@gep.reak.bme.hu:/tmp/
Password:
valami.txt        100%    0      0.0KB/s    00:00
```

Akkor a következő történik. Az `scp` parancs mondja meg hogy itt most titkosított fájlküldés történik. a `~` jel jelöli, hogy a saját `home` könyvtárunkból szeretnénk az azon belüli `mappa` mappában elhelyezett `valami.txt` fájlt mozgatni. A célgép a `gep.reak.bme.hu`, a felhasználó akinek a nevében cselekszünk, a `user`. A gépnév végén egy kettőspont választja el a gépnevet a távoli gépen kívánt elérési úttól, ami most a `/tmp/` mappa, tehát a távoli

gépen a `/tmp/valami.txt` helyre fog kerülni a fájl. Amennyiben lehagyjuk a kettőspont után a `/` jelet, úgy az elérési út nem a gyökérkönyvtártól, hanem a `user` felhasználó `home` könyvtárától lesz értendő. A parancs kiadása után a rendszer elkéri jelszavunkat (ha nincs kulcsos azonosítónk) majd egy progress bar jelzi a folyamat állapotát: a %-os feldolgozottságot, a küldött adat mennyiségét, a mindenkori sávszélességet, és az eltelt időt.

Az egyszerűség kedvéért az adatleolvasó– és mentő programokban ez az opció be van építve, nem szükséges külön másolgatnunk kézzel. A default elérési út egy olyan gép olyan mappája, amelyre általában van kulcsos azonosítás a gate1 adott user accountjából, így nem szükséges jelszó megadása, az adatlekérők automatikusan működnek. Biztonsági okokból ezt az elérési utat és accountot ebbe a dokumentációba nem írhatom bele, de a programok futás közben kiírják ezt az utat, illetve a forráskódból is visszakereshető.

1.2.4. A szükséges mappastruktúra

Figyelem 1: ez a szekció folyamatos bővítés alatt áll.
Figyelem 2: Tervben van véve egy telepítőscript elkészítése, amely létrehozza a szükséges mappastruktúrát.

MATLAB:
Matlab használata esetén a programjainkat egy `routines` mappába tegyük, és a `routines` mappa mellé készítsünk el egy `raw_data` mappát, ahova a lementett adatokat tesszük, illetve ahol az adatokat a többi program keresni fogja. Ezen mappák hiánya esetén a programok elkészítik maguknak a szükséges mappákat amennyiben nem léteznek, hogy a lefagyást és adatvesztést elkerüljék. Hogy biztosan egységes legyen a mappastruktúra, jobb kézzel elkészíteni a szükséges mappákat. A MATLAB-ot a `routines` mappából indítsuk el.

A `pog` account-ban a `papp` könytárban ez a mappastruktúra létre van már hozva. A `~/papp/` mappán belüli `routines` mappa ezen SXR programokból a fejlesztés ideje alatt tartalmazza a mindenkori legfrissebb verziót, a `raw_data` mappa pedig a már korábban lementett több gigabyte-nyi adatot.

1.2.5. MATLAB és IDL indítása

Akár MATLAB akár IDL programban dolgozunk, szükséges ezek munkamappájának beállítása: a mappa, ahol a futtatni kívánt fájlokat keresik.

Ennek legegyszerűbb módja, ha indítás előtt benavigálunk a megfelelő mappába, ahova MATLAB vagy IDL programjainkat tettük, és onnan indítjuk a MATLAB-ot vagy az IDL-t. Így az adott indítási mappa lesz a program alapértelmezett munkakönyvtára.

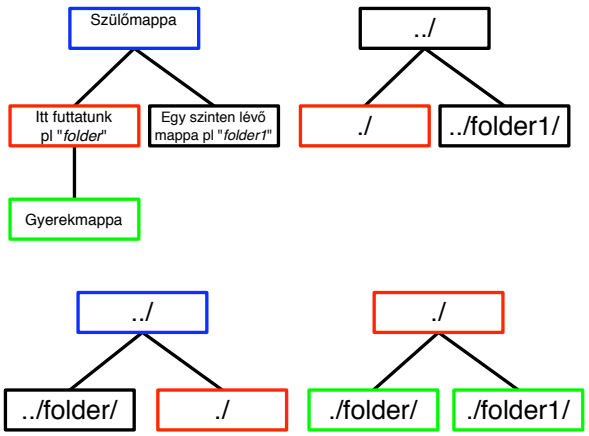
Az IPP rendszerén a konzolból kiadott `matlab` illetve `idl` parancsokkal tudjuk szöveges módban indítani ezeket a programokat. Az IPP-ben defaultnak beállított MATLAB és IDL programverziók kompatibilisek ezzel az SXR programcsomaggal.

A továbbiakban a példákban a MATLAB konzolt `>>`, az IDL konzolt pedig `IDL>` fogja jelölni.

1.3. Jelölések

1.3.1. Mappajelölések

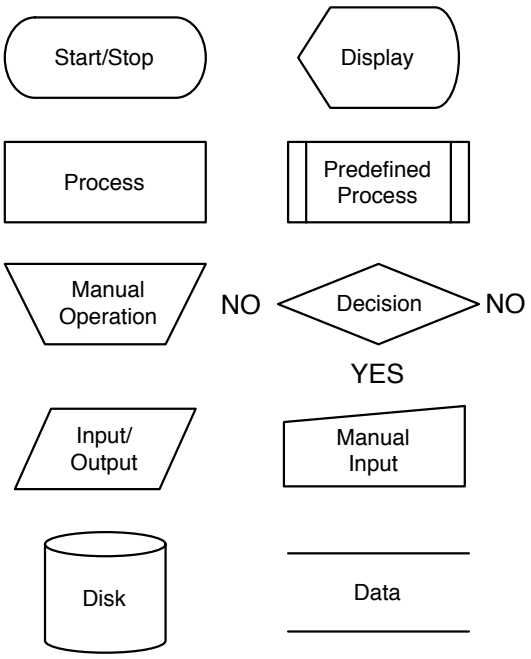
A mappaszerkezetet az a jelölésrendszer jelöli a dokumentumban, ami a 1.1. ábrán látható. Az ábrán piros jelöli az aktuális mappát, jelölése `'./'`. **Fekete jelöli a vele egy szinten lévő testvérmappát**, jelölése `'../mappanev/'`. Kék jelöli az anya/szülőmappát, jelölése `'./'` , és zöld jelöli a gyerekmappát, ami az aktuális mappából nyílik, ennek jelölése `'./mappanev/'`



1.1. ábra. Mappaszerkezet jelölése különböző szituációkban.

1.3.2. Folyamatábrák

A folyamatábrák jelölésrendszere a 1.2. ábrán látható. A folyamatábrákon mindig lefelé vagy jobbra haladunk.



1.2. ábra. A folyamatábrák jelölésrendszere

Start/stop: A program indulása/leállása.
Display: Információ kiírása a felhasználó felé.
Process: Valamilyen művelet végrehajtása.
Predefined process: valamilyen egyéb program hívása.
Decision: Választás. Lefelé van a logikai igen, oldalra a logikai nem.
Manual operation: Kézi beavatkozás szükséges (pl. fájlmozgatás).
Input/Output: Változók beolvasása vagy visszaadása.
Manual Input: Felhasználó általi adatmegadás szükséges.
Disk: Adatírás merevlemezre.
Data: Adatolvasás merevlemezről.

2. fejezet

Adatlekérés - MATLAB

Az adatlekérő programok csak az IPP rendszerében futtatva működnek! Ezért futtatásukhoz be kell lépni valamilyen IPP hálózatba kapcsolt gépre, a programokat fel kell másolni oda pl. egy becsatolt afs meghajtón keresztül. A MATLAB programokat egy `routines` nevű mappába tegyük, és ezen mappával egy szinten hozzunk létre egy `raw_data` mappát amibe majd a `.mat` adatok kerülnek. Ha ez a mappa nincs meg, a program elkészíti nekünk. Rendelkeznünk kell érvényes token-el is, amit a `klog` parancs kiadása és az afs jelszavunk (login jelszó) megadása után kapunk meg. Az AUG-nál az adateléréshez általánosan a következő adatokra van szükség:

- Lövésszám (továbbiakban **shot**) pl. `20975`, stb.
- Az adott diagnosztikai csatorna (egy Mirnov szonda, egy SXR csatorna, stb) neve (továbbiakban **channel**) pl. `'J_053'`, stb. A lágy röntgen diagnosztika leírásánál találjuk az elérhető csatornaneveket: https://www.aug.ipp.mpg.de/~vgi/SXR.html
- Azon adatfeldolgozó számítógép neve amire az adott `channel` csatlakoztatva van (továbbiakban **diag**) pl `'SXA'`, `'MHA'`, stb.
- Az adatlekérés kezdeti ideje (továbbiakban **t_begin**) az adott lövésen belül, lebegőpontos számmal másodpercben megadva pl `5.5432`, stb.
- Az adatlekérés végének ideje (továbbiakban **t_end**) az adott lövésen belül, lebegőpontos számmal másodpercben megadva pl `5.672`, stb.

2.1. gp_sxr_savedata.m (SXR mentés)

2.1.1. Rendeltetés

Beolvassa, fájlba menti és opcionálisan egy távoli gépre másolja a megadott `shot` számú lövésből a `channel`-ben megadott SXR csatorna vagy kamera adatát a [`t_start, t_end`] időintervallumban. Amennyiben az `scp` kapcsoló értéke 1, úgy távoli gépre másolja ssh kapcsolaton az adatfájlt a megadott `remote` elérési útra. Ha `remote` nincs megadva, a program a default helyre másol. A program futtatása MATLAB-ból történik, lásd 1.2.

2.1.2. Használat

```
gp_sxr_savedata(shot,channel,t_start,t_end,scp,[remote])
```

Bemenetek:

- `shot`: a lövésszám egész számként, pl. `20975`
- `channel`: az adott csatorna kódja sztringként, pl. `'J_053'`. Nem javasolt, de lehetséges egyben egész csatornacsoportok letöltése is. Az ilyenkor szükséges bemenet illetve annak eredménye megtalálható a program kódjában.
- `t_start`: A kérdéses időintervallum eleje lebegőpontos számként másodpercben megadva, pl. `0.142`
- `t_end`: A kérdéses időintervallum eleje lebegőpontos számként másodpercben megadva, pl. `5.476`
- `scp`: Ha értéke `1`, akkor automatikus másolás történik a `remote` változóban meghatározott értékre. Ha nincs megadva, az aktiválja a bemeneti felületet! Default értéke `0`.
- `[remote]`: A távoli másoláskor a felhasználónév, távoli gép és távoli célmappa megadása, unix konvencióval. Értéke egy darab sztring.Például: `'user@gep:/mappa1/mappa2/'` Értékét csak akkor kell megadni, ha felül szeretnénk írni a default értéket. Hiánya nem aktiválja a bemeneti felületet.

Ha az első 5 bemenet valamelyike hiányzik, a program aktiválja a bemeneti felületet. Ezesetben bekéri az összes adatot. `scp` default értéke `0`, elég `enter`-t ütni rá. `remote` bekérdezése csak akkor történik meg ha `scp=1`. `remote`

kiírt default értékének elfogadásához szintén elegendő `enter`-t ütni.

Kimenet:
A program fájlt készít kimenetként. A fájl helye a programhoz viszonyítva a `../raw_data/` mappa (1.3.1 fejezet). Ezt a mappát létre kell hoznunk, hiányában a program létrehozza magának. A fájlnév a következő: `shot_channel_tstart_tend.mat`. Például: `20975_J_053_4.12_4.15.mat`
A fájlnév elnevezése csak az egyediséget szolgálja, a későbbiek során a fájlnévben tárolt információ nem kerül felhasználásra, mindig a `.mat` fájlból olvasódnak ki az adatok. Tehát a fájlnév szabadon megváltoztatható, ez a programok működését semmilyen módon nem befolyásolja.

Kimeneti adatformátum:

A `.mat` fájlba egy `sxr_g` struktúra kerül mentésre. Ezt egy példán keresztül érthetjük meg a legjobban.

```
sxr_g =
        t: [1x20001 double]
       tN: 'Time'
      tpD: 's'
        y: [1x20001 double]
       yN: 'J_053'
      ypD: 'V           '
     diag: 'SXB'
     shot: 20975
    start: 0
      end: 0.0100
```

Az `sxr_g.t` illetve `sxr_g.y` mezők tartalmazzák az idő- illetve adatvektort `1 X hossz` formátumban, duplapontosságú lebegőpontos számként. A `tN` és `yN` illetve `tpD` és `ypD` mezők a `t` és `y` mezők nevei illetve dimenziói. A `sxr_g.diag` mező tartalmazza a diagnosztikai számítógép nevét, továbbá visszakapjuk a `shot` lövésszámot és a választott időintervallum `t_start` és `t_end` határait. Ez a struktúra előnyös, mert tetszőleges számú mezőt adhatunk hozzá és még mindig csak egy változóval kell törődnünk.
Néhány példa az ezzel történő bánásra:

1. Átadjuk az idővektort a struktúrából egy `t` vektroba:

   ```
   >> t=sxr_g.t;
   ```

```
>> whos t
  Name      Size          Bytes  Class

  t         1x2001        16008  double array

Grand total is 2001 elements using 16008 bytes
```

2. Kiiratjuk a csatorna nevét:

```
>> disp(sxr_g.yN)
J_053
```

3. Ha csatornacsoportot mentettünk le, akkor `sxr_g(1,N)` formátumú lesz, ahol `N` a csatornák száma a csoportban. Ekkor következőképpen néz ki az adat:

```
>> gp_sxr_savedata(20975,'J',0,0.001,0)
(...)
Succesfully saved ../raw_data/20975_J_0_0.001.mat
>> load('../raw_data/20975_J_0_0.001.mat')
>> whos sxr_g
  Name        Size          Bytes  Class

  sxr_g       1x15         495470  struct array

Grand total is 60600 elements using 495470 bytes
>> disp(sxr_g(1,5).yN)
J_050
```

Példák

1. Programhívás direkt adatokkal:

```
>> gp_sxr_savedata(20975,'J_053',0,0.001,1, \\
         'user2@gep.reak.bme.hu:asdex/raw_data/')
Loading SXR signal(s)...
Requesting shot 20975 of AUGD:SXB 20975
SF2ML  : load shot 20975 from 0.000000 [s] to 0.001000 [s]
1 tracks with 2001 values/track (indicees : 1 - 2001)
SF2ML : voila, data ready.
================================
```

```
Saving data... (Please be patient)
Succesfully saved ../raw_data/20975_J_053_0_0.001.mat
Secure copying file to user2@gep.reak.bme.hu:asdex/raw_data/
(Your ssh password may be needed)
Password:
20975_J_053_0_0.001.mat     100%   32KB  32.4KB/s   00:00
```

2. Programhívás adatok nélkül:

```
>> gp_sxr_savedata
 Please load the information about the requested shot
Shot number:  20975
Camera name: J_053
Begin of the time interval:  0
End   of the time interval:  0.001
Copy to remote machine automatically? ([0]/1)  1
Remote machine [default is user@gep.reak.bme.hu:asdex/raw_data/]:
Loading SXR signal(s)...
Requesting shot 20975 of AUGD:SXB 20975
SF2ML  : load shot 20975 from 0.000000 [s] to 0.001000 [s]
1 tracks with 2001 values/track (indicees : 1 - 2001)
SF2ML : voila, data ready.
==================================
Saving data... (Please be patient)
Succesfully saved ../raw_data/20975_J_053_0_0.001.mat
Secure copying file to user@gep.reak.bme.hu:asdex/raw_data/
(Your ssh password may be needed)
20975_J_053_0_0.001.mat     100%   32KB  32.4KB/s   00:00
```

3. Programhívás hiányos adatokkal: Ugyanúgy, mint 2.2.2.

4. Programhívás hibás adatokkal: Ugyanúgy mint 2.2.2.

2.1.3. Működés

A program működését a 2.1. ábra alapján érthetjük meg. A program indulása után beveszi az adatokat (már amennyit kap).

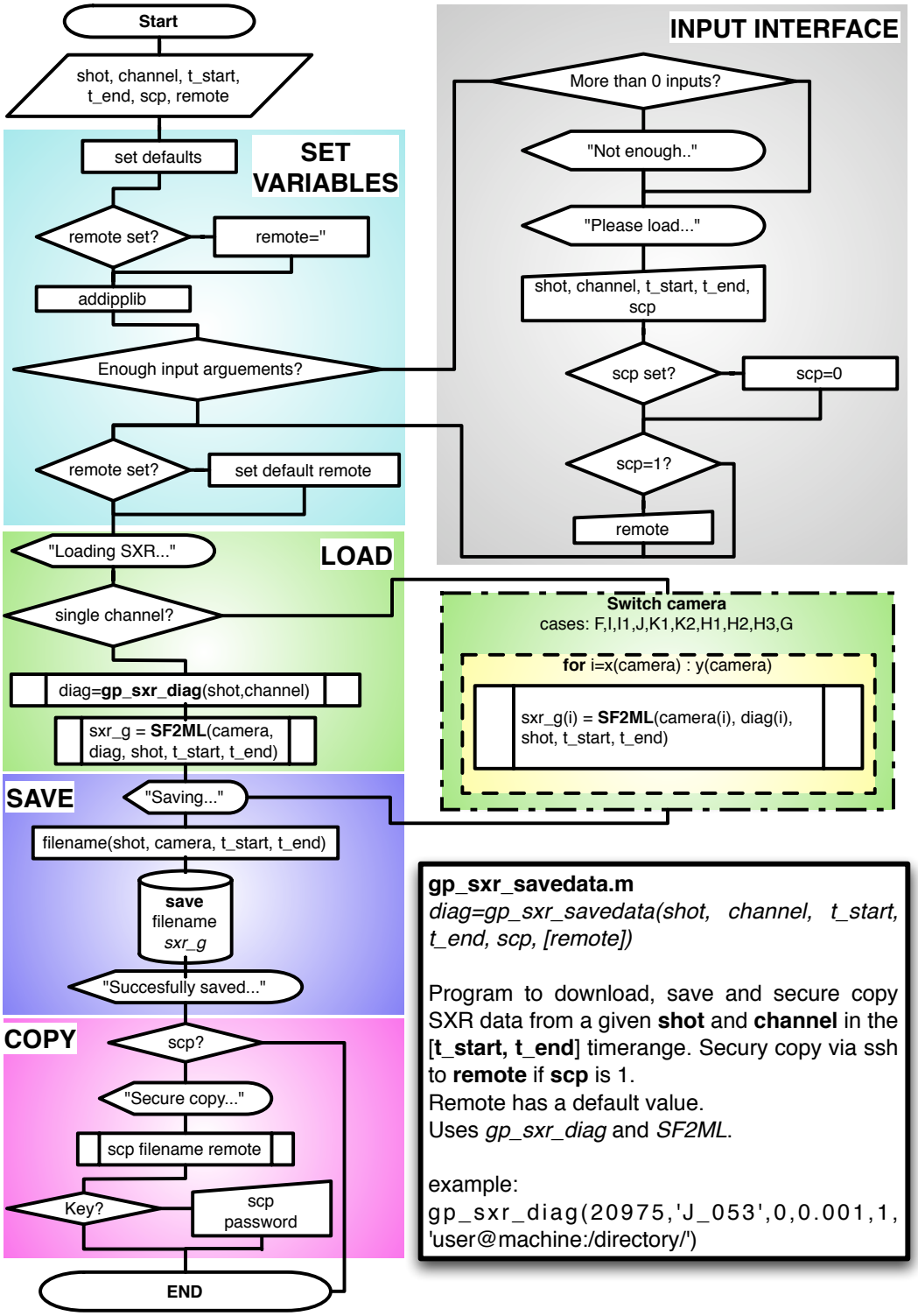
A `SET VARIABLES` rész:
Beállításra kerülnek a default értékek, pl `remote`-é. Ha a remote értéke nincsen megadva akkor deklaráljuk. Ezután hozzáadásra kerül a standard IPP

könyvtár az addipplib segítségével, ez kell majd a későbbiek során az SF2ML hívásához. Amennyiben nincs meg az 5 kötelező input adat, úgy aktiválódik a bemeneti interfész (később). Ha megvan az 5 kötelező adat de remote értéke hiányzúk, úgy az felveszi default értékét.

AZ INPUT INTERFACE rész:
Amennyiben van, csak nem elégséges input, egy figyelmeztetést kapunk. Ezek után a program szól hogy aktiválta a bemenetet. Ezt követően kerül beolvasásra az 5 kötelező argumentum, egyenként, kérdéses formában. A korábban leírtak szerint kell bevinni őket. Amennyiben scp-re csak egy enter-t ütünk úgy ő üres lesz, ezért átadjuk neki a default 0 értéket. Amennyiben scp értékadása után értéke 1, úgy a program rákérdez a remote értékére. Itt bevihető új, üresen hagyva felveszi default értékét.

A LOAD rész:
Egy figyelmeztetés után a program ellenőrzi, hogy egy csatornát vagy egy csatornacsoportot kívánunk-e beolvasni. Ha csatornacsoportot, akkor a gp_sxr_diag függvény segítségével beállítja diag értékét, majd meghívja az SF2ML függvényt amivel az adatot letölti az AUG rendszeréből (LOAD) és az sxr_g struktúrába rakja le.

A switch camera rész:
Nem javasolt, de lehetséges egyszerre több csatorna jelét lementeni, ezesetben sxr_g egy vektor lesz, 1xcsatornak dimenzióval, az egyes mezők ugyanezek mint fentebb. A csatornacsoportok az egyes kamerák egyforma mintavételezésű részeit választják ki. Használata nem javasolt, mert egyetlen csatorna hibája a teljes adatelérést meghiúsítja, továbbá végeredményeként nagyon nagy fájlok keletkeznek. Programozástechnikailag egy switch figyeli a bemeneti channel értékét és amennyiben ez egy előre megadott értéket vesz fel (pl. 'G') akkor az a rész aktiválódik. Itt egy for ciklus megfelelően indexelve megy végig az SF2ML rutin többszöri hívásán. A diag értékek kézzel vannak beírva az egyes csoportokhoz.
A SAVING rész:
Itt egy figyelmeztető üzenet és a filename fájlnév összerakása után (formája shot_channel_tstart_tend.mat, például: 20975_J_053_4.12_4.15.mat) történik meg a fájlírás. A fájl a programhoz viszonyított ../raw_data/ mappába (lásd 1.3.1. fejezet) kerül kiírásra, tartalma az sxr_g struktúra. A mappát jobb a biztos alapon hozzuk létre, de hiánya esetén a program elkészíti magának. A sikeres fájlírás eltarthat egy ideig, végét üzenet jelzi.



2.1. ábra. A `gp_sxr_diag.m` program blokkvázlata.

A `COPY` rész:
Amennyiben `scp=1`, úgy automatikusan megtörténik az adatnak az IPP rendszeréből távoli számítógépre történő másolása. Erre egy üzenet figyelmeztet. A program a unix rendszer `scp` parancsát hívja meg és a korábban letett fájlt a `remote` változóban meghatározott helyre másolja. Amennyiben ssh kulcsos azonosítás van az IPP-s afs accountunk és a `remote`-ban megnevezett távoli hely között, úgy a másolás automatikus. Amennyiben nem, szükséges megadni a `remote` helyen érvényes jelszavunkat. `remote` értékét praktikus olyan helyre beállítani, ahova van ssh kulcsos hozzáférésünk.

Részletesebb magyarázatok találhatók a program forráskódjában. A program rövid segítsége elérhető a MATLAB konzolból is a `help gp_sxr_savedata` parancs kiadásával.

2.2. gp_sxr_diag.m (Diagnosztika megkeresése)

2.2.1. Rendeltetés

A `shot` és `channel` adatokhoz automatikusan hozzárendeli a `diag` értéket.

Ez a függvény a 2.1-ban leírt `gp_sxr_savedata` program belső függvénye, önmagában nem szoktuk használni. Adatlekérés programozásakor lehet hasznos.

Az, hogy egy adott csatorna melyik diagnosztikai számítógépre van dugva az SXR esetén nem triviális, az adatrögzítést több mint egy tucat számítógép vezérli. A csatornakiosztás az idők során változott/változhat. A diagnosztika honlapjáról[1] a csatornakiosztás táblázatos formában elérhető, de ezt használni nehézkes és nem biztos hogy aktuális.

2.2.2. Használat

```
diag=gp_sxr_diag(shot,channel)
```

Bemenetek:

- `shot`: a lövésszám egész számként, pl. `20975`
- `channel`: az adott csatorna kódja sztringként, pl. `'J_053'`

[1] https://www.aug.ipp.mpg.de/~vgi/SXR.html, login szükséges!

Ha bármelyik bemenet hiányzik, a program aktiválja az adatbeviteli felületet, tehát a program input paraméterek nélkül is hívható, lásd később.

Kimenet:

- `diag`: a kérdéses diagnosztikai számítógép neve, egy három karakter hosszú sztring, pl `'SXB'`

Példák:

1. Közvetlen futtatás a változók megadásával

```
>> diag=gp_sxr_diag(20975,'J_053');disp(diag);
SXB
```

2. Futtatás interaktívan. A `20975` és `J_053` adatokat futás közben kellett begépelni, `enter`-t ütve minden sor végén. A csatorna nevét `'` jelek nélkül kell gépelni.

```
>> diag=gp_sxr_diag; disp(diag)
 Please load the information about the requested DIAG
Shot number:  20975
Camera name: J_053
SXB
```

3. Futtatás hiányos adatokkal. Ha nincs elég bemenet, a program aktiválja a beviteli felületet.

```
>> diag=gp_sxr_diag(20975); disp(diag)
Not enough input arguements (<2), activating input interface...
 Please load the information about the requested DIAG
Shot number:  20975
Camera name: J_053
SXB
```

4. Hibás bemenetek kezelése. A `shot` értéke csak szám lehet, ellenkező esetben hibaüzenetet kapunk. Paraméterként megadott adat esetén a futás leáll. A beviteli felület használatakor újrakéri az adatot. Ha a `channel` értékének nem a megfelelő módon, vagy invalid értéket adunk meg, akkor hibaüzenetet kapunk. Ekkor a program kimenete `ERR` lesz.

```
>> diag=gp_sxr_diag; disp(diag)
 Please load the information about the requested DIAG
Shot number:  asdf
??? Undefined function or variable 'asdf'.

Shot number:  20975
Camera name: asdf
Error in WhichSX while getting diagnostic name \\
for -asdf- in shot -20975-!
Channel was possibly not connected.
ERR
```

2.2.3. Működés

Az automatizált **diag(shot,channel)** összerendelést az IPP számítógépes rendszerén belül elérhető `WhichSX` program szolgáltatja. Ez egy shell script, melynek szintaxisa `WhichSX shot channel`. Ez kerül hívásra a MATLAB alól. A működéséhez kell, hogy legyen érvényes token-ünk az afs rendszerre, ezt a `klog` segítségével tehetjük meg. Lássunk egy példát hívásra:

```
pog@seaug2 1) klog
Password:
pog@seaug2 2) WhichSX 20975 J_053

Now starting read_diags...
---------------------------
diags: SXA  SXB  SXC  SXD  SXF  SXG  SXH  SXI  SXJ  SXK  SXL  SXM  SXN

Now starting find_common_nr...
---------------------------
commonshotnr: 20807
WARNING: highest address is: 128 not equal to number of channels!

****************************************
signal: J_053   address: 27     Diag: SXB
****************************************
```

Az itt szereplő Diag értéke az amire szükségünk van. Ezt a sort `grep` parancs segítségével választjuk le a kimenetről:

```
pog@seaug2 4) WhichSX 20975 J_053 | grep Diag
signal: J_053   address: 27     Diag: SXB
```

Még le kell vágni a sor végéről a megfelelő három karaktert. A sor vége egy üres karakter, ezért az üres karakterektől meg kell szabadulni a `tr -d " "` parancs segítségével:

```
pog@seaug2 12) WhichSX 20975 J_053 | grep Diag | tr -d " "
signal:J_053    address:27      Diag:SXB
```

A levágást a `cut` paranccsal csinálom meg. A `:` karaktert elválasztónak használva a 4. mezőt választom ki:

```
pog@seaug2 11) WhichSX 20975 J_053 | grep Diag | tr -d " " | cut -d: -f4
SXB
```

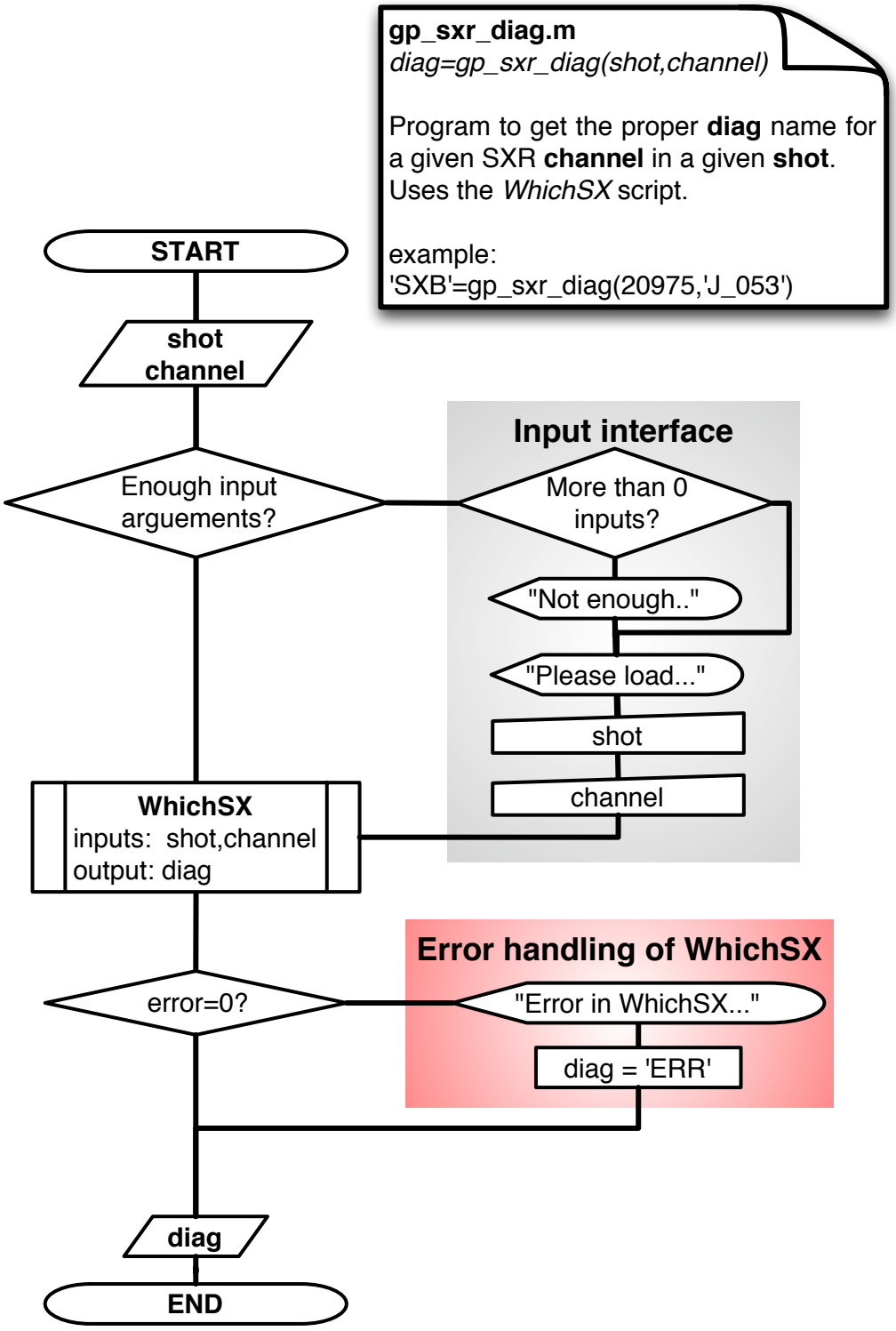
Ezt a shell parancsot kell meghívni a MATLAB-ból. Amennyiben a csatorna nincs bekötve, a WhichSX program kimenete `channel not connected`, ami a fenti formájú hívás esetén üres kimenetet ad vissza. A MATLAB-ban shell parancsokat a `unix` utasítással hívom meg és veszem vissza a script kimenetét. A MATLAB egy üres karakterként hozzűfűzi a kimenethez a sortörést amit a parancsértelmező tesz a parancs végén. Így a korábban végéről levágott szünet jel visszakerül, mikor a MATLAB átveszi a kimenetet. Ezért az `echo -n` paranccsal adom vissza az előző script végeredményét, ez elnyomja a sortörés karaktert. A végső parancs így néz ki:

```
pog@seaug2 12) echo -n ‘WhichSX 20975 J_053 | \\
   grep Diag | tr -d " " | cut -d: -f4‘
SXBpog@seaug2 13)
```

A sort a láthatóság miatt tört, a parancssorba az itt sortörést jelölő \\ nélkül, egyben kellene begépelni. Az így kapott kimenet összecsúszik a parancsértelmező nevével, de a MATLAB így fogja jól átvenni.

A `gp_sxr_diag` program működését mutatja a 2.2. ábra. Program indulásakor megadható a két bemenet, de nem kötelező. Amennyiben nincs elégséges bemenet, úgy aktiválódik a beviteli felület, és manuálisan kell megadni a `shot` és `channel` értékeket. Egy üzenet figyelmeztet ha van ugyan bemenet, de nem elégséges. Ezután kerül meghívásra a `WhichSX`. Amennyiben a `WhichSX` hívásakor hiba történt vagy a kimenet üres, akkor a program hibaüzenetet ad és `diag` az `ERR` (sztring) értéket veszi fel. Ezután visszaadásra kerül a `diag` értéke.

Részletes magyarázat található a program forráskódjában. A program rövid segítsége elérhető a konzolból is a `help gp_sxr_diag` parancs kiadásával.



2.2. ábra. A `gp_sxr_diag.m` program blokkvázlata.

2.3. gp_ece_savedata.m (ECE mentés)

2.3.1. Rendeltetés

Beolvassa, fájlba menti és opcionálisan egy távoli gépre másolja a megadott `shot` számú lövésből az összes ECE csatorna adatát a [`t_start, t_end`] időintervallumban. Amennyiben az `scp` kapcsoló értéke 1, úgy távoli gépre másolja ssh kapcsolaton az adatfájlt a megadott `remote` elérési útra. Ha `remote` nincs megadva, a program a default helyre másol. A program futtatása MATLAB-ból történik, lásd 1.2.

2.3.2. Ismert hibák

Nem teljesen tisztázott, de vélhetőleg az `SF2ML`-es adatlehívó hibájából az ECE adat időnként értelmezhetetlen: egyes csatornák azonosan 0 adatot tartalmaznak, még más csatornák 10^{23} nagysűgrendű értékeket, de elektronvoltban. ECE lehívásra a tapasztalatok szerint az IDL beolvasó megbízhatóbb.

2.3.3. Használat

```
gp_ece_savedata(shot,t_start,t_end,scp,[remote])
```

Bemenetek:

- `shot`: a lövésszám egész számként, pl. `20975`
- `t_start`: A kérdéses időintervallum eleje lebegőpontos számként másodpercben megadva, pl. `0.142`
- `t_end`: A kérdéses időintervallum eleje lebegőpontos számként másodpercben megadva, pl. `5.476`
- `scp`: Ha értéke `1`, akkor automatikus másolás történik a `remote` változóban meghatározott értékre. Ha nincs megadva, az aktiválja a bemeneti felületet! Default értéke `0`.
- `[remote]`: A távoli másoláskor a felhasználónév, távoli gép és távoli célmappa megadása, unix konvencióval. Értéke egy darab sztring.Például: `'user@gep:/mappa1/mappa2/'` Értékét csak akkor kell megadni, ha felül szeretnénk írni a default értéket. Hiánya nem aktiválja a bemeneti felületet.

Ha az első 4 bemenet valamelyike hiányzik, a program aktiválja a bemeneti felületet. Ezesetben bekéri az összes adatot. `scp` default értéke `0`, elég `enter`-t ütni rá. `remote` bekérdezése csak akkor történik meg ha `scp=1`. `remote` kiírt default értékének elfogadásához szintén elegendő `enter`-t ütni.

Kimenet:
A program fájlt készít kimenetként. A fájl helye a programhoz viszonyítva a `../raw_data/` mappa (1.3.1 fejezet). Ezt a mappát létre kell hoznunk, hiányában a program létrehozza magának. A fájlnév a következő: `shot_ECE_tstart_tend.mat`. Például: `20975_ECE_0_8.mat` A fájlnév elnevezése csak az egyediséget szolgálja, a későbbiek során a fájlnévben tárolt információ nem kerül felhasználásra, mindig a `.mat` fájlból olvasódnak ki az adatok. Tehát a fájlnév szabadon megváltoztatható, ez a programok működését semmilyen módon nem befolyásolja.

Kimeneti adatformátum:

A `.mat` fájlba három struktúra kerül mentésre. Ezt egy példán keresztül érthetjük meg a legjobban.

```
>> load('../raw_data/20975_ECE_0_8.mat')
>> whos
  Name       Size                Bytes  Class     Attributes

  ece        1x1             111937460  struct
  ece_R      1x1                 74812  struct
  ece_z      1x1                 74812  struct
```

Betöltjük az adatfájlt, majd kiiratjuk a változókat, akkor látjuk hogy három új változónk van, mindhárom struktúra. Nézzük ezeket egyenként:

```
>> ece
ece =
        t: [1x229372 double]
       tN: 'time-A'
      tpD: 's'
        y: [60x229372 double]
       yN: [60x10 char]
      ypD: [60x12 char]
     diag: 'CEC'
     shot: 20975
    start: 0
      end: 8
```

Az `ece.t` illetve `ece.y` mezők tartalmazzák az idő- illetve adatvektorokat `1 * hossz` illetve `csatorna * hossz` formátumban, duplapontosságú lebegőpontos számként. A példában 60 csatornánk van, ezért az `ece.y` egy `60 * hossz` méretű mátrix, minden sor egy-egy csatorna adatát tartalmazza.

A `tN` és `yN`; illetve `tpD` és `ypD` mezők a `t` és `y` mezők nevei; illetve dimenziói. Az egyes csatornák nevei és dimenziói külön kerülnek elmetésre, így pl. `ece.yN` a példában egy `60 X 10`-es sztring mátrix.

Az `ece.diag` mező tartalmazza a diagnosztikai számítógép nevét (itt `'CEC'`), továbbá visszakapjuk a `shot` lövésszámot és a választott időintervallum `start` és `end` határait. Ez a struktúra előnyös, mert tetszőleges számú mezőt adhatunk hozzá és még mindig csak egy változóval kell törődnünk. Néhány példa az ezzel történő bánásra:

1. Átadjuk az idővektort a struktúrából egy `t` vektroba:

```
>> t=ece.t; whos t;
  Name      Size              Bytes  Class     Attributes

  t         1x229372        1834976  double
```

2. Kiiratjuk a 25-ös csatorna nevét:

```
>> disp(ece.yN(25,:))
Trad-A(25)
```

3. Vektorba mentjük a 13-as csatorna jelét:

```
>> data=ece.y(13,:); whos data
  Name      Size              Bytes  Class     Attributes

  data      1x229372        1834976  double
```

Most nézzük a két következő struktúrát:

```
>> ece_R
ece_R =
        t: [1x146 double]
       tN: 'rztime'
      tpD: 's'
        y: [60x146 double]
       yN: [60x7 char]
```

```
      ypD: [60x12 char]
     diag: 'CEC'
     shot: 20975
    start: 0
      end: 8

>> ece_z
ece_z =
        t: [1x146 double]
       tN: 'rztime'
      tpD: 's'
        y: [60x146 double]
       yN: [60x7 char]
      ypD: [60x12 char]
     diag: 'CEC'
     shot: 20975
    start: 0
      end: 8
```

Az itteni két struktúra nagyban hasonlít az előzőekre. Ez a két változó tárolja a csatornák pozícióját az idő függvényében, de jelentősen rosszabb időfelbontással, mint magát az adatot. A `diag, shot, start, end` mezők ugyanazok. A `ece_z.t` és `ece_R.t` ugyanúgy az idővektor, `ece_z.y` és `ece_R.y` pedig maga pozíciómátrix. Minden egyes csatorna R és z koordinátáját adja meg. Mértékegysége `ece_z.ypD` és `ece_R.ypD`-ben van tárolva, itt méter. `ece_z.yN` és `ece_R.yN` az adatmátrix egyes soraihoz tartozó adat neve. A mezők elérése teljesen hasonló mint a `ece` változó esetén.

A csatornapozíciók automatikusa mentésre kerülnek, és használhatjuk is megjelenítésre. Viszont a csatornapozíciók megjelenítéséhez praktikusabb az IPP gépein elérhető `diaggeom` programot használni.

Példák

1. Programhívás direkt adatokkal:

```
>> gp_ece_savedata(20975,0,8,1, \\
'geri@deep.reak.bme.hu:asdex/raw_data/')
Loading ECE signals...
Requesting shot 20975(3) of AUGD:CEC 20975(3)
SF2ML  : load shot 20975 from 0.000000 [s] to 7.339872 [s]
60 tracks with 229372 values/track (indicees : 2 - 229373)
```

```
SF2ML warning in ddcsgrp at signal group Trad-A : \\
see details in /tmp/fort.pid (pid=process id)
SF2ML : voila, data ready.
Requesting shot 20975(3) of AUGD:CEC 20975(3)
SF2ML  : load shot 20975 from 0.100000 [s] to 7.350000 [s]
60 tracks with 146 values/track (indicees : 1 - 146)

SF2ML warning in ddcsgrp at signal group z-A : \\
see details in /tmp/fort.pid (pid=process id)
SF2ML : voila, data ready.
Requesting shot 20975(3) of AUGD:CEC 20975(3)
SF2ML  : load shot 20975 from 0.100000 [s] to 7.350000 [s]
60 tracks with 146 values/track (indicees : 1 - 146)

SF2ML warning in ddcsgrp at signal group R-A : \\
see details in /tmp/fort.pid (pid=process id)
SF2ML : voila, data ready.
================================
Saving data... (Please be patient)
Succesfully saved ../raw_data/20975_ECE_0_8.mat
Secure copying file to user2@deep.reak.bme.hu:asdex/raw_data/
(Your ssh password may be needed)
Password:
20975_ECE_0_8.mat          100%  107MB   8.9MB/s   00:12
10.33user 2.32sys 1:01.98 20.4%
```

Meghívjuk a programot. Ezek után kapunk egy `'Loading...'` figyelmeztetést. Az ezt követő információkat az `SF2ML` adja vissza, jelen esetben visszajelzések és egy apró figyelmeztetés. A három adathívás végét a `====` vonal jelzi. Ekkor történik meg az adatmentés, majd a másolás távoli gépre (kellett SSH jelszó is).

2. Programhívás adatok nélkül:

```
>> gp_ece_savedata
 Please load the information about the requested shot
Shot number:  20975
Begin of the time interval:  0
End   of the time interval:  8
Copy to remote machine automatically? ([0]/1)  1
Remote machine \\
```

```
[default is user@deep.reak.bme.hu:asdex/raw_data/, \\
to accept, hit return]:

Loading ECE signals...
Requesting shot 20975(3) of AUGD:CEC 20975(3)
SF2ML  : load shot 20975 from 0.000000 [s] to 7.339872 [s]
60 tracks with 229372 values/track (indicees : 2 - 229373)

SF2ML warning in ddcsgrp at signal group Trad-A : \\
see details in /tmp/fort.pid (pid=process id)
SF2ML : voila, data ready.
Requesting shot 20975(3) of AUGD:CEC 20975(3)
SF2ML  : load shot 20975 from 0.100000 [s] to 7.350000 [s]
60 tracks with 146 values/track (indicees : 1 - 146)

SF2ML warning in ddcsgrp at signal group z-A : \\
see details in /tmp/fort.pid (pid=process id)
SF2ML : voila, data ready.
Requesting shot 20975(3) of AUGD:CEC 20975(3)
SF2ML  : load shot 20975 from 0.100000 [s] to 7.350000 [s]
60 tracks with 146 values/track (indicees : 1 - 146)

SF2ML warning in ddcsgrp at signal group R-A : \\
see details in /tmp/fort.pid (pid=process id)
SF2ML : voila, data ready.
================================
Saving data... (Please be patient)
Succesfully saved ../raw_data/20975_ECE_0_8.mat
Secure copying file to user@deep.reak.bme.hu:asdex/raw_data/
(Your ssh password may be needed)
20975_ECE_0_8.mat            100%  107MB   8.9MB/s   00:12
10.44user 2.32sys 0:12.48 102.2%
>>
```

Itt ugyanaz történik mint előző esetben, csak aktiválódott a beviteli felület, és a default távoli gépre nem kellett jelszó.

3. Programhívás hiányos adatokkal: Ugyanúgy, mint 2.2.2.

4. Programhívás hibás adatokkal: Ugyanúgy mint 2.2.2.

2.3.4. Működés

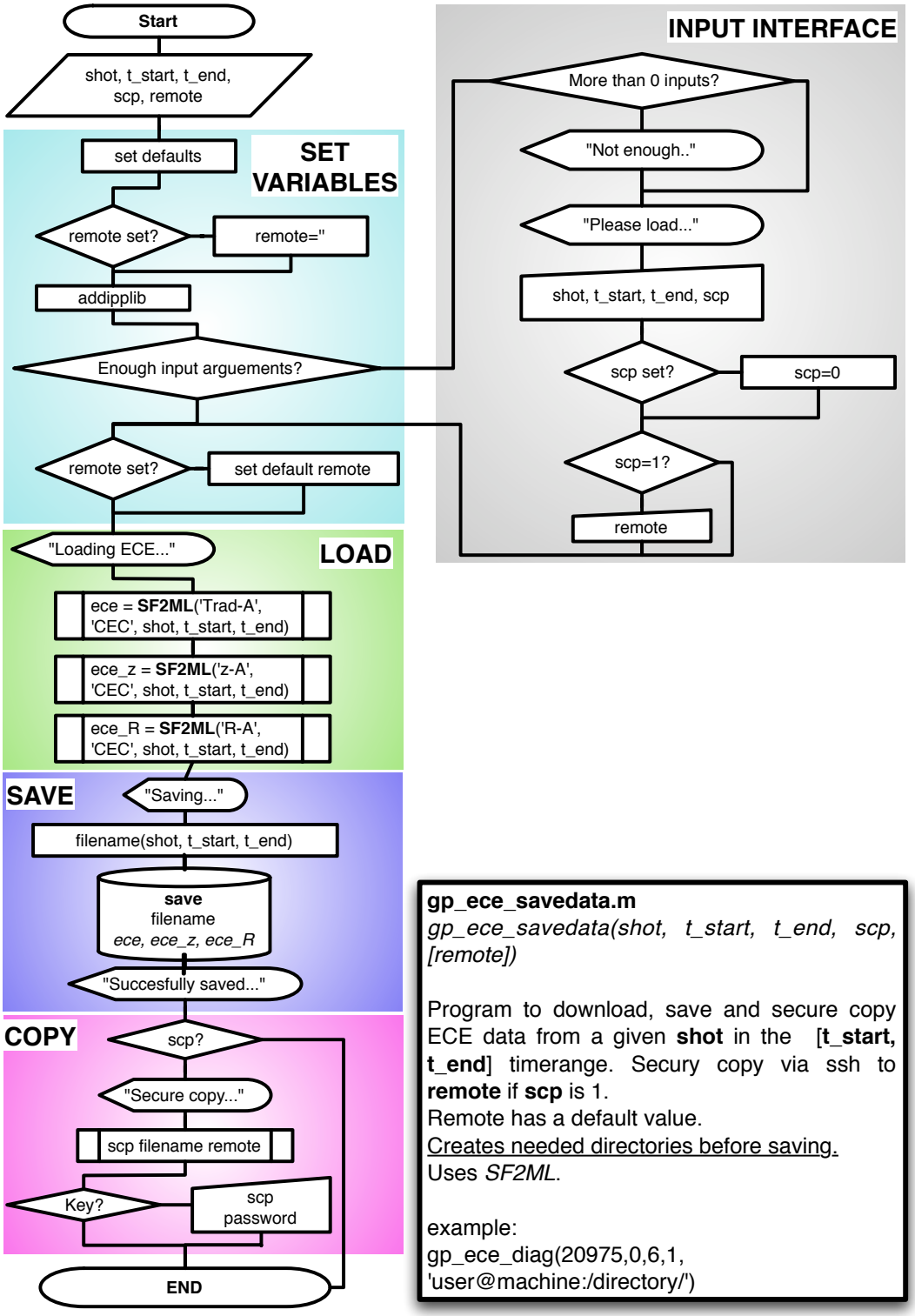
A program működését a 2.3. ábra alapján érthetjük meg. A program indulása után beveszi az adatokat (már amennyit kap).

A `SET VARIABLES` rész:
Beállításra kerülnek a default értékek, pl `remote`-é. Ha a remote értéke nincsen megadva akkor deklaráljuk. Ezután hozzáadásra kerül a standard IPP könyvtár az `addipplib` segítségével, ez kell majd a későbbiek során az `SF2ML` hívásához. Amennyiben nincs meg a 4 kötelező input adat, úgy aktiválódik a bemeneti interfész (később). Ha megvan a 4 kötelező adat de `remote` értéke hiányzik, úgy az felveszi default értékét.

AZ `INPUT INTERFACE` rész:
Amennyiben van, csak nem elégséges input, egy figyelmeztetést kapunk. Ezek után a program szól hogy aktiválta a bemenetet. Ezt követően kerül beolvasásra a 4 kötelező argumentum, egyenként, kérdéses formában. A korábban leírtak szerint kell bevinni őket. Amennyiben `scp`-re csak egy `enter`-t ütünk úgy ő üres lesz, ezért átadjuk neki a default 0 értéket. Amennyiben `scp` értékadása után értéke `1`, úgy a program rákérdez a `remote` értékére. Itt bevihető új, üresen hagyva felveszi default értékét.

A `LOAD` rész:
A program figyelmeztet hogy kezdődik az adatbehívás, majd az `SF2ML` rutint használva lehív három adatstruktúrát. Az `ece` maga az adat, az `ece_R` és `ece_z` az egyes csatornák pozíciója az idő függvényében.

A `SAVING` rész:
Itt egy figyelmeztető üzenet és a `filename` fájlnév összerakása után (formája `shot_ECE_tstart_tend.mat`, például: `20975_ECE_0_8.mat`) történik meg a fájlírás. A fájl a programhoz viszonyított `../raw_data/` mappába (lásd 1.3.1. fejezet) kerül kiírásra, tartalma három struktúra. A mappát jobb a biztos alapon hozzuk létre, de hiánya esetén a program elkészíti magának. A sikeres fájlírás eltarthat egy ideig, végét üzenet jelzi.



2.3. ábra. A `gp_ece_diag.m` program blokkvázlata.

A `COPY` rész:
Amennyiben `scp=1`, úgy automatikusan megtörténik az adatnak az IPP rendszeréből távoli számítógépre történő másolása. Erre egy üzenet figyelmeztet. A program a unix rendszer `scp` parancsát hívja meg és a korábban letett fájlt a `remote` változóban meghatározott helyre másolja. Amennyiben ssh kulcsos azonosítás van az IPP-s afs accountunk és a `remote`-ban megnevezett távoli hely között, úgy a másolás automatikus. Amennyiben nem, szükséges megadni a `remote` helyen érvényes jelszavunkat. `remote` értékét praktikus olyan helyre beállítani, ahova van ssh kulcsos hozzáférésünk.

Részletesebb magyarázatok találhatók a program forráskódjában. A program rövid segítsége elérhető a MATLAB konzolból is a `help gp_ece_savedata` parancs kiadásával.

3. fejezet

Adatlekérés - IDL

A sztenderd `get_rawsignal.pro` alá vannak integrálva az egyes IDL olvasó modulok, esetenként saját vagy külső függvényeket használva.

Az adatlekérés ebben az esetben is csak az IPP hálózatán belülről futtatva érhető el. Az IDL programokat az '/afs/ippgarching.mpg.de/home/p/pog/idl/' mappában tároljuk. A programok kimenete valamint a lekért adatok az ezen belüli 'data/output/' mappába töltődnek.

A sztenderd `get_rawsignal.pro` alá vannak integrálva az egyes IDL olvasó modulok, esetenként saját vagy külső függvényeket használva. Ha a program megtalálja a beolvasni kívánt adatot a 'data' mappában, akkor betölti onnan, ha nem találja, akkor letölti az adatot ide, majd betölti.

3.1. SXR

3.1.1. Használat

Az adatlekérés a már említett `get_rawsignal.pro` rutin segítségével történik.

Paraméterek:

- shot: lövésszám
- signal name: az sxr adatok esetében: 'AUG_SXR/' + a csatorna neve. Például 'AUG_SXR/J_053'
- timeax: ebbe a vektorba adja vissza az időtengelyt
- data: ebbe a vektorba adja vissza az adatokat
- trange (opcionális): az időintervallum, amin belül az adatokat betölti. FONTOS: a trange kapcsolónak csak a timeax illetve data vektorokban

vissza adott adatokra van hatása, akár használjuk, akár nem, mindig letölti a teljes lövés alatt rögzített SXR adatot.

- movedata (opcionális): a movedata=1 opció segítségével automatikusan áthelyezhetjük a letöltött adatokat a Deep szerverre, a '/home/data/asdex/raw_data/' mappa alá. Alapértelmezett beállítás esetén nem helyezi át az adatot, hanem a ' /idl/data/' mappába tölti le.

Példa adatletöltésre:

```
IDL> get_rawsignal, 20975, 'AUG_SXR/J_053', timeax, data, $
trange=[2,6], movedata=1
```

A fenti példa esetén az adatot 2-6 másodpercig adja vissza a timeax illetve data vektorokban, a teljes letöltött adatot pedig áthelyezi a Deep szerverre.

3.1.2. Működés

A `get_rawsignal.pro` rutin az SXR adatok lekérésére egy másik rutint, az `ma_sxr_savedata.pro` -t használja.

ma_sxr_savedata.pro A rutin hívása:

```
ma_sxr_savedata, shot, channel, [trange], $
[save_path] [/scp], [remote], [/plot], $
[/no_correction], [data], [time]
```

Bemenetek:

- shot: lövésszám
- channel: csatorna név (pl.: 'J_053')
- trange (opcionális): idő intervallum, alapértelmezett: teljes lövés
- save_path (opcionális): letöltött adatok helye, alapértelmezett: ' /idl/data/'
- /scp (opcionális): adatok áthelyezése távoli számítógépre
- remote(opcionális): a távoli számítógép címe, alapértelmezett: deep.reak.bme.hu:/home/dat
- /plot (opcionális): ábrázolja a letöltött adatokat
- /no_correction (opcionális): Hotlink error javítása nélkül tölti le az adatokat
- data (opcionális): visszaadja az adatokat

- time (opcionális): visszaadja az időtengelyt

A lementett fájlba a következő adatok kerülnek:

- data: SXR adat
- starttime: kezdő időpont
- endtime: végső időpont
- sampletime: mintavételi idő, két szomszédos adatpont közti időtartam
- info: információ a lövésről (Az IPP-ben elérhető SXinfo nevű programból)

3.1.3. Függvények

Fűrészfog összeomlás kereső rutin

A rutin neve: `ma_findcrash.pro`,a célja, hogy megkeresse a fűrészfog összeomlások idejét az adott lövésben.

Működés: A fűrészfog összeomlások spektrogramjait figyelve a legtöbb esetben találhatunk olyan frekvenciasávot, melyben csak az összeomlásokkor látható egy-egy szélessávú összetevő, egyébként az összeomlások között, a frekvenciasáv üres. Ha ezen a frekvenciasávon kiszámítjuk a sávteljesítményt, azt találjuk, hogy a sávteljesítménynek éles maximumai vannak az egyes összeomlások idején.

A fűrészfog összeomlás kereső rutin így működik: kiszámítja a sávteljesítményt a megadott frekvenciasávon, majd megkeresi azoknak a maximumoknak a helyét, melyeknél a sávteljesítmény nagyobb, mint egy bizonyos korlát. A korlát értékét úgy számítja ki, hogy egy állítható faktorral megszorozza a jel átlagát.

Ha a megadott SXR jel hosszabb, mint 0.5 másodperc, akkor a program szétbontja 0.5 másodperc hosszú szakaszokra a jelet és mindegyiken különkülön elvégzi a keresést.

Programhívás:

```
ma_findcrash, shot, channel [, datapath] [,trange] $
[,frange] [,threshold]
```

Bemenetek:

- shot: lövésszám
- channel: csatorna név

- datapath: az SXR adat helye, alapértelmezett: './data/'
- trange: idő intervallum, amin belül keresi az összeomlásokat, alapértelmezett: teljes lövés
- frange: frekvenciasáv, amin a sávteljesítményt számolja a program. FONTOS: ennek egy olyan frekvenciasávnak kell lennie, ami csak az összeomlásokkor tapasztalt szélessávú komponenseket tartalmazza, egyébként pedig üres. Alapértelmezett frekvenciasáv: 1-3 kHz.
- threshold: korlát = jel átlaga * threshold, alapértelmezett: threshold=10.

Kimenetek:

- Az SXR adat 0.5 másodperc hosszú spektrogramjai.
- Az adott frekvenciasávon számított sávteljesítmények.
- A futás végén a program a képernyőre kiírja a talált összeomlások idejét.

FONTOS: A kapott eredményt célszerű ellenőrizni oly módon, hogy a futás során legyártott spektrogramokat végignézzük, ellenőrizzük, hogy valóban üres-e a megadott frekvenciasáv illetve valóban fűrészfog összeomlás okozta-e a szélessávú összetevőt. Az utóbbi főként a jel elejére illetve végére vonatkozik, ahol gyakran előfordulnak szélessávú zajok. Abban, hogy mennyire üres az adott frekvenciasáv a sávteljesítmények segítségével is könnyen ellenőrizhető. Egy gyors ellenőrzési módszer: megszámolom a spektrogramon a fűrészfog összeomlásokat, ha ugyan ennyi összeomlást talált a program az adott intervallumban, akkor valószínűleg jól működött.

3.2. ECE

3.2.1. Használat

Az ASDEX Upgrade ECE jeleinek beolvasója az SXR beolvasóhoz hasonlóan a `get_rawsignal.pro` -ba van integrálva. Az adatok elérése is hasonlóan működik, új adatokat az ASDEX AFS rendszerén belül futtatva tölthetünk le.

Bemenetek:

- shot: lövésszám

- channel: csatorna, AUG ECE esetében ez 01-60 -ig terjed. Pl.: $'AUG_ECE/05'$
- time: idő vektor
- data: adat vektor
- movedata: az adatokat áthelyezi távoli számítógépre, a movedata=1 hívással a Deep-re helyezi. Alapértelmezetten nem helyezi át, hanem a *'data'* mappába teszi.

Példa:

```
get_rawsignal, 20975, 'AUG_ECE/25', time, data, movedata=1
```

A program letöltéskor mind a 60 csatorna adatát lekéri, ezeket 1 fájlba írja be, majd ebből a fájlból olvassa be a kért csatornát. A fájlon belül egy ECE nevű struktúrában tárolja az adatokat, ebben a struktúrában szerepelnek olyan adatok is, amiket a `get_rawsignal.pro` nem olvas be.

3.2.2. Működés

3.2.3. Függvények

3.3. Mágneses adatok

3.3.1. Használat

3.3.2. Működés

3.3.3. Függvények