Manual auxiliar PYHDUST

Daniel Moser

25 de novembro de 2014

Capítulo 1

Pol

Em reuniões em nov/2014 com Moser, Bednarski e Alex, ficou decidido:

- Cada estrela (alvo ou padrão) terá um PREFIXO único de observação. Essa lista é gerada da seguinte forma: copia-se a coluna no arquivo *planejamento_LNA.xls* num arquivo de texto, que será lido pelas rotinas (arquivo pyhdust/pol/alvos.txt). Destaca-se que a identificação do alvo é feita desta forma, não importando o prefixo dos arquivos fits.

- A estrutura de diretórios "local" é *noite/prefixoalvo_n*. Assim, todo sub-diretório da noite é considerado um alvo, exceto *calib*. Caso uma estrela tenha sido observada em mais de uma sequencia, adicionar _ *n*, onde pode ser um número (e.g., 2) ou um descritivo (e.g., *a*2). Tudo o que está em subdiretórios dos alvos é ignorado.

- A estrutura de diretórios será *dadospuros/noite/alvo* e *reduzidos/noite/alvo*. O script *pur2red* limpa o resultado da redução dos dados puros. O script *red2pur* copia os resultados da redução em reduzidos para a pasta dos dados puros.

- O critério para a escolha do(s) arquivos *out(s)* será o de minimizar os erros em blocos de 16 ou 8+8 posições, mantendo o máximo de pontos independentes possível. Os detalhes estarão no manual das rotinas.

- As rotinas PYTHON propostas são as seguintes: (i) *genStdLog*, (ii) *genObjLog*, (iii) *genStdDat*, (iv) *genTarget*, (v) *pur2red*, (vi) *red2pur* e (vii) *listNights*. Consultar o manual das rotinas para a lista completa e detalhes.

- É possivel demonstrar que $P = \sqrt{Q^2 + P^2}$ e $\sigma_P = \sigma_{QU}$.

Passos da redução:

1. Obtem-se a lista de noites do objeto (ou via página Beacon, ou via rotina *listNights* caso tenha acesso a pasta *puros*).

2. Gera-se os arquivos de calibração-padrão (rotina IRAF *calib*).

3. Reduz-se as padrões da noite no IRAF. Roda-se *genStdLog*. A rotina imprime alertas de dados de padrões não reduzidos. Colunas da saída de *genStdLog*: MJD|target|filt|calc|out|flags| (um *out* por filtro/padrão).

4. Reduz-se os alvos da noite (IRAF). Roda-se *genObjLog* e imprime alertas de dados não reduzidos/estrelas fora da nomenclatura. Colunas da saída de *genObjLog*: MJD|target|filt|calc|out_n|flags| (pode haver mais de um *out* por filtro/alvo; registro em nova linha).

5. Roda-se *genStdDat*. A rotina faz: (i) olha na saída de *genObjLog* os filtros/calcitas dos alvos utilizados, (ii) e espera encontrar as padrões correspondentes; (iii) caso não encontre uma padrão, faz interpolação (de filtros). Se faltou para toda a calcita, pede para o usuário fazer a referência a saída de outro *genStdLog*.

6. Reduz-se todas as noites (de interesse). Roda-se *genTarget* para um, múltiplos ou todos os alvos. Procura noite à noite o alvo na saída de *genObjLog*, calibrado com *genStdDat* da noite correspondente (detalhe: nesta configuração, os arquivos *outs* serão lindos neste momento. Ou seja, o usuário depois dos passos 1, 2 e 3, ainda precisa ter acesso a estes arquivos). Neste momento, o usuário confirma qual ângulo de referência da padrão será utilizado (padrão é o "publicado"). Caso exista mais de uma padrão para o filtro/calcita, faz uma média do $\Delta\theta$. Colunas da saída de *genTarget* (saída individual para cada estrela):
MJD|noite|filt|calc|ang.ref|del.th|P|Q|U|th|sigP|sigth|flags|.

Outras definições sugeridas:

- No caso de um alvo observado com duas calcitas (como padrões), não colocar no mesmo diretório. Isso complica muito o desflexibiliza muito as rotinas que gerenciam os artivos **.out*. A regra é criar uma nova pasta com o sufixo "_a2" contendo os arquivos fits. A raiz destes arquivos é arbitrária.

- Além disso, outra situação é necessária para se abrir outra pasta: ao se iniciar uma nova sequência temporal.

- Nomenclatura de observação *target_ a0_ g1_ f.fits* onde $a0$ e $g1$ são argumentos opcionais (calcita e ganho/config. do CCD) e f é o filtro. Na

ausência destes parâmetros, assume-se valor padrão (ou lê-se no *header* dos fits).

- Os avisos em tela *Warning* avisa o usuário que uma informação foi registrada, mas ela não é a otimizada. Por exemplo, na ausência de uma das versões de redução (.1 ou .2) ou do agrupamento de posições da lâmina (08 ou 16).

- Os avisos em tela *Error* avisa o usuário que uma informação foi perdida. Dependendo o erro, o script será interrompido (não gerando nenhum output parcial).

Definições das flags.

0 As flags são cumulativas. Se estiver 0, significa que tudo ocorreu bem (nenhuma decisão ou aproximação no processo de redução foi feita). Se ocorreram, terá uma ou mais das flags abaixo.

pA Estrela sem padrão para calibração de *dth* (output é salvo mesmo assim).

pB A correção *dth* foi feita usando uma estrela de outra noite.

pC A correção *dth* foi feita usando duas ou mais estrelas.

Capítulo 2

Filters

Transmissao dos filtros UBVRI (Bessel 1990).

Arquivos pegos em
http://pan-starrs.ifa.hawaii.edu/project/people/kaiser/
filter_transfer_functions/johnsoncousins/

Os filtros JHK foram pegos do sítio do SOAR.

Capítulo 3

Photlines

- **generate_hdust_hlines.pro**: programa que calcula as linhas de transicao do H, chama o *synspec* e lê o fluxo com modelos do Tlusty ou Kurucz. A estrutura em λ é mantida a mesma para salvar memória/tempo.
- **Importante!** O *synspec* tem 2 modos básicos de execução: (i) completo (modos 0 e 1 do synplot); e (ii) só contínuo+linhas H (modo 2 do synplot). A rotina IDL acima só funciona para o modo 2, i.e., para as linhas do Hidrogenio. Para linhas genéricas, usar rotina adequada do PYHDUST. Isto porque os pontos da linha são adicionadas ao contínuo, e perde-se o controle da resolução (na entrada no HDUST exige-se mesmo tamanho para a criação dos vetores *gfortran*).
- ~~Pelo que me lembro, *Nlower* e *Nupper* são as séries que estarão inclusas na saída (ex., 1=Lyman, 3=Paschen). *ilow* é o número de transições incluídas em cada série (ex. ilow=4 haverá α à δ).~~
- *ilow* é a série mínima (ex., 1=Lyman). *Nlower*Nupper* é o número total de linhas. *Nlower* e *Nupper* é uma combinação das transições.
- ~~No caso de linha não-do hidrogênio, usar o programa **generate_hdust_line.pro**. No futuro, estes 2 arquivos *.pro estarão no PYHDUST.~~
- Note que a inclusão de química além de H e He altera a entrada do *synspec* (ver arquivos *atmos.5*).

3.1 Formato arquivo XDR

O formato é o seguinte:

```
Nupper  Nlower  ilow    npts    DeltaV(km/s)
npts    Nup     Nlow    #referente a transicao do H
(vetor lambda com npts)
npts    Nup2     Nlow2    #referente a outra transicao do H
(vetor lambda com npts)
...
#Nlower*Nupper vezes
...
teff    logg
(fluxo com npts, da 1a transicao)
(fluxo com npts, da 2a transicao)
teff2    logg2
teff    logg
(fluxo com npts, da 1a transicao)
(fluxo com npts, da 2a transicao)
...
#Total de modelos de atmos.
```

Formato para linhas genéricas é idêntico, com exceção que $Nlower = ilow = 1$.

3.2 Install

Funcionando só para linhas do H.

```
#install
rm rotin3 synspec49
cd sources/
cd synspec49/
rm *.o
make
mv synspec49 ../../
cd ../rotin3
gfortran rotin3.f -o ../../rotin3
cd ../../

#running manually
idl
.r generate_hdust_lines.pro
.r synsplot49.pro
generate_hdust_lines.pro
```

```
#running at background
idl < gen.idl > out.txt 2> err.txt &

#synplot49.show.pro == displays the plot on IDL window
#synplot49.noshow.pro == do not displays the plot on IDL window

# ATM_lines.dat == default output name
# Convert to the XDR format:
python generate_hdust_lines.py kurucz_lines.dat
```

3.3 Cookbook

```
Hi Jon,

Here's a brief cookbook to get started with synspec.
Since you're interested in hot stars, I'll use the
model atmosphere computed by Hubeny & Lanz for O and
B-type stars.

Synspec is f77, but what I have used mainly is an
IDL wrapper around it which prepares the input files and
reads the output. Of course synspec can be used totally
independent of the wrapper (synplot.pro).

The source code that you will need to do the tests
here are synspec and synplot. Both are here
ftp://hebe.as.utexas.edu/pub/callende/jon/synspec.tar.gz
A slightly older version of synspec is available from
http://nova.astro.umd.edu/Synspec43/synspec.html
where you can also find the manual and input data.
To install, compile synspec (use -fno-automatic if
using g77 or gfortran), and place the IDL code in your idl path.

The necessary input for computing spectra of hot stars will be
model atmospheres from
http://nova.astro.umd.edu/
atomic data (model atoms) from
http://nova.astro.umd.edu/Tlusty2002/tlusty-frames-data.html
a linelist for the UV or visible from
```

```
http://nova.astro.umd.edu/Synspec43/synspec-frames-data.html
or, for the H-band from
http://hebe.as.utexas.edu/apogee/

Let's do an example for a 15000 solar-metallicity giant:
1) download BGmodels_v2.tar from the tlusty site
2) unpack and place in your local dir the files:
BG15000g175v2.5 -> control file
BG15000g175v2.7 -> model atmosphere (atmospheric structure
and level populations)
BG15000g175v2.nst -> non-standard (auxiliary) parameter file

3) copy/rename/link BG15000g175v2.nst to nst (or change
the name of the aux. file in BG15000g175v2.5)
4) get your model atoms from the tlusty site. you will
need all the files named in BG15000g175v2.5, but rather than
downloading one by one you can get them all in a tar ball
from the same web page: atom_BS06.tar. unpack them
5) get your linelist (I'm using gfVIS99.dat for the example below)
and link it or rename it as fort.19
6) open idl and go for it!
IDL> .r synplot
% Compiled module: SYNPLOT.
IDL> synplot,0,0,0,atmos='BG15000g175v2',wstart=3000.,wend=3500.,x,y
total equivalent width:       3807.00  milliangstrom
% Compiled module: SET_XY.

If everything has worked well, you will end up with two
arrays with wavelengths and fluxes.
IDL> help,x,y
X               FLOAT     = Array[1, 52091]
Y               FLOAT     = Array[1, 52091]

Hope this helps, but let me know if  you run into trouble.
Cheers,

Carlos
```