UNIVERSIDADE DO VALE DO PARAÍBA

INSTITUTO DE PESQUISA E DESENVOLVIMENTO

Astrofísica Observacional

Modulo IRAF#2 – Espectroscopia

Matheus Soares Palhares

5 de novembro de 2015

São José dos Campos, SP

1. INTRODUÇÃO

A espectroscopia é uma das ferramentas fundamentais para a astronomia, e com o aumento de detectores digitais em espectrógrafos aumentou muito a capacidade do astrônomo realizar essa tarefa. Porém, com este aumento da capacidade, começou a se preocupar com as ferramentas utilizadas para redução destes dados, precisando aperfeiçoar técnicas para este trabalho com tarefas mais sofisticadas. Uma maneira de aperfeiçoar essas tarefas é o desenvolvimento de algoritmos capazes de realizar essa redução de espectros de uma maneira fidedigna. Com toda essa inovação tecnológica, a redução de um espectro ficou muito mais rápida do que era antes (as placas fotográficas eram digitalizadas por um fotômetro e só depois eram reduzidas). Este manual, mostra uma das ferramentas mais comuns utilizadas para a redução de espectros.

2. INSTALAÇÃO DO IRAF

A instalação do IRAF pode ser feita de diversas maneiras. Através de instalações manuais encontradas na internet (por exemplo em http://astr.tohoku.ac.jp/~akhlaghi/irafinstall.html) ou diretas (http://ssb.stsci.edu/ureka/ , por exemplo). Neste manual a instalação seguiu o método direto pelo UREKA em ambiente Linux, que consiste de baixar um arquivo .sh e rodá-lo na pasta de instalação.

Após essa instalação podemos executar o IRAF de duas maneiras

1. Direto pelo terminal, executando os comandos ur_setup e cl
2. Pelo terminal do UREKA, basta procura-lo depois de aberto usa o comando cl.

3. DADOS

Os dados utilizados para este manual foram obtidos pelo espectrógrafo *Goodman* instalado no telescópio SOAR no Chile. Estes dados fazem parte de um projeto por busca de variáveis cataclísmicas magnéticas do qual o autor faz parte. Para este manual utilizamos os dados do objeto V348 Pav (SSS110526).

A identificação dos arquivos de dados foi feita a partir de uma rotina desenvolvida pelo grupo conhecida como obs_log.cl, essa rotina tem por finalidade a extração de campos do cabeçalho das imagens como, por exemplo, o nome do arquivo (FILE), tempo de exposição (EXPTIME), a massa de ar (AIRMASS), o objeto observado (OBJECT) e o tipo de observação (OBSTYPE). Estes dois últimos (OBJECT e OBSTYPE) são os campos que iremos usar neste momento. O tipo de observação pode ser bias, flat, lâmpadas de comparação, e os objetos de estudo (tanto estrela espectrofotométricas e objetos de ciência). O campo OBJECT define o que cada imagem contém.

Após o reconhecimento dos dados aconselho fazer uma lista com cada tipo de objeto para facilitar os demais passos. As listas podem ser feitas direto pelo IRAF com o comando ls utilizando a seguinte sintaxe para lista:

ls -1 nome_do_objeto > nome_da_lista

Agora podemos seguir para a redução dos dados.

4. REDUÇÃO DOS DADOS

Os processos de redução são os mesmos utilizados para a fotometria exemplificada no manual anterior feito pelo Dr. Will Robson M. Rocha.

Essa redução tem o objetivo de eliminar assinaturas causadas por determinadas características físicas e operacionais presentes nos CCDs. Estas assinaturas interferem na contagem de fótons, então são indispensáveis para a análise dos dados. As assinaturas são os BIAS (nível de carga pré alocado no CCD) e FLATS (sensibilidade de cada pixel do CCD). As etapas para estas correções serão descritas abaixo. Além dessas correções efetuamos o TRIM (corte nas imagens) para eliminar regiões de OVERSCAN bem como sua correção.

4.1 Correção de OVERSCAN e TRIM

A região de OVERSCAN fica localizada na borda da imagem obtida com o CCD, e aparece devido à sua própria eletrônica quando ele é feito o processo de leitura. Assim, o nível de pixels na região do OVERSCAN dá uma média do sinal introduzido pela leitura do CCD, o que também é chamado de nível de OVERSCAN. Dessa forma, após ser computado, ele deve ser removido. A remoção é feita fazendo um corte na imagem e preservando apenas a região de interesse. Esse corte é chamado de TRIM e a região que tiramos foi da linha 10 até a 2055 e da coluna 57 até 826. O TRIM é feito pelo CCDPROC do pacote NOAO.IMRED.CCDRED do IRAF. Para efetuarmos o TRIM e o OVERSCAN quando necessário utilizamos o CCDPROC de acordo com os parâmetros abaixo. Essa tarefa é usada em todas as imagens, só após o TRIM começamos a correção por BIAS e FLAT.

```
   TASK = ccdproc

images  =  @todos  List of CCD images to correct
(output =        ) List of output CCD images
(ccdtype=        ) CCD image type to correct
(max_cac=       0) Maximum image caching memory (in Mbytes)
(noproc =      no) List processing steps only?

(fixpix =      no) Fix bad CCD lines and columns?
(oversca=      no) Apply overscan strip correction?
(trim   =     yes) Trim the image?
(zerocor=     yes) Apply zero level correction?
(darkcor=      no) Apply dark count correction?
(flatcor=     yes) Apply flat field correction?
(illumco=      no) Apply illumination correction?
(fringec=      no) Apply fringe correction?
(readcor=      no) Convert zero level image to readout correction?
(scancor=      no) Convert flat field image to scan correction?

(readaxi=    line) Read out axis (column|line)
(fixfile=        ) File describing the bad lines and columns
(biassec=        ) Overscan strip image section
(trimsec= [10:2055,57:826]) Trim data section
(zero   = zero.fits) Zero level calibration image
(dark   =        ) Dark count calibration image
(flat   = nflat.fits) Flat field images
(illum  =        ) Illumination correction images
(fringe =        ) Fringe correction images
```

```
(minrepl=      1.) Minimum flat field value
(scantyp= shortscan) Scan type (shortscan|longscan)
(nscan  =       1) Number of short scan lines

(interac=      no) Fit overscan interactively?
(functio= legendre) Fitting function
(order  =       1) Number of polynomial terms or spline pieces
(sample =       *) Sample points to fit
(naverag=       1) Number of sample points to combine
(niterat=       1) Number of rejection iterations
(low_rej=      3.) Low sigma rejection factor
(high_re=      3.) High sigma rejection factor
(grow   =      0.) Rejection growing radius
(mode   =      q)
```

4.2 Correção de BIAS

O BIAS está associado ao ruído de leitura do CCD. Após as cargas serem armazenadas no poço de potencial, depois da exposição do telescópio, os elétrons são transportados para um amplificador e para um conversor analógico digital, que transforma cada carga em um pacote de dados será lido pelo computador. Essa leitura é traduzida como um determinado número de contagens. Assim temos um ganho determinado pela razão de elétrons por ADU (Analogical Digital Unity). A questão é que esse ganho não é o mesmo para todos os CCDs, e, portanto, isso está associado ao ruído de leitura. Então, são feitas exposições curtas (com o menor tempo possível do CCD) com o obturador da câmera fechado para que seja computado apenas o ruído. Na prática, são feitas em torno de 50 a 100 imagens BIAS, e então é obtida uma imagem com o ruído médio. Esse processo faz com que tenhamos uma relação sinal ruído melhor para a análise de imagens. Dessa forma, as imagens dos campos são subtraídas do ruído médio proveniente do CCD. Para os dados que utilizamos foram feitos 50 BIAS. A maneira obter a imagem média é através da tarefa ZEROCOMBINE localizada no mesmo pacote anterior. Os parâmetros que utilizamos estão abaixo.

```
TASK = zerocombine

input   = @lista_bias  List of zero level images to combine
(output = zero.fits) Output zero level name
(combine=   average) Type of combine operation
(reject = avsigclip) Type of rejection
(ccdtype=          ) CCD image type to combine
(process=        no) Process images before combining?
(delete =        no) Delete input images after combining?
(clobber=        no) Clobber existing output image?
(scale  =      none) Image scaling
(statsec=          ) Image section for computing statistics
(nlow   =         0) minmax: Number of low pixels to reject
(nhigh  =         1) minmax: Number of high pixels to reject
(nkeep  =     1) Minimum to keep (pos) or maximum to reject (neg)
(mclip  =       yes) Use median in sigma clipping algorithms?
(lsigma =        3.) Lower sigma clipping factor
(hsigma =        3.) Upper sigma clipping factor
(rdnoise=      3.99) ccdclip: CCD readout noise (electrons)
(gain   =      2.06) ccdclip: CCD gain (electrons/DN)
(snoise =        0.) ccdclip: Sensitivity noise (fraction)
```

```
(pclip  =      -0.5) pclip: Percentile clipping parameter
(blank  =        0.) Value if there are no pixels
(mode   =         q)
```

Feito a imagem média pelo ZEROCOMBINE agora precisamos subtrair das imagens de ciência e dos FLATs, para isso utilizamos o CCDPROC alterando o campo de BIAS (exemplo do CCDPROC nos parâmetros anteriores). Estamos quase no fim das correções, vamos agora para os FLATs.

4.3 Correção de FLAT

O FLAT é uma correção utilizada para calibrar a sensibilidade de cada pixel do CCD, pois os CCDs possuem um ganho diferente entre si, o que gera respostas diferentes para uma mesma intensidade de radiação. Este efeito é multiplicativo no valor das contagens. A imagem de calibração de FLAT é obtida através de imagens de uma fonte mais luminosa que os objetos astronômicos, para que haja uma relação sinal-ruído alta, e exposições rápidas para evitar a saturação do CCD. Estas imagens devem possuir uma iluminação uniforme em cada pixel. Há diferentes maneiras para obter as imagens de FLAT. Os procedimentos mais comuns são exposições do interior da cúpula do telescópio ou de uma tela montada em seu interior (flat de cúpula), ambos iluminados por uma lâmpada; ou então por exposições do céu durante o crepúsculo (skyflat) ou ainda uma iluminação por lâmpada interna ao instrumento. Como a sensibilidade dos pixels também depende da faixa espectral é necessário que as imagens de FLAT sejam feitas com os mesmos filtros utilizados nas observações de ciência. São feitas, novamente, exposições suficientes para obter uma boa estatística (para este trabalho foram feitas 50 exposições), após isso, combina-se as imagens para obter uma imagem média do valor encontrado para o FLAT. As imagens de ciência são, ao final, divididas pelo FLAT médio. Neste trabalho optou-se por usar o FLAT internos de lâmpada de quartzo. Para combinar os FLATs utilizamos a tarefa FLATCOMBINE com os parâmetros descritos abaixo.

```
TASK = flatcombine

input = @lista_flat  List of flat field images to combine
(output =  flat.fits) Output flat field root name
(combine=    average) Type of combine operation
(reject =  avsigclip) Type of rejection
(ccdtype=           ) CCD image type to combine
(process=         no) Process images before combining?
(subsets=         no) Combine images by subset parameter?
(delete =         no) Delete input images after combining?
(clobber=         no) Clobber existing output image?
(scale  =       mode) Image scaling
(statsec=           ) Image section for computing statistics
(nlow   =          3) minmax: Number of low pixels to reject
(nhigh  =          3) minmax: Number of high pixels to reject
(nkeep  =     1) Minimum to keep (pos) or maximum to reject (neg)
(mclip  =        yes) Use median in sigma clipping algorithms?
(lsigma =         3.) Lower sigma clipping factor
(hsigma =         3.) Upper sigma clipping factor
(rdnoise=       3.99) ccdclip: CCD readout noise (electrons)
(gain   =       2.06) ccdclip: CCD gain (electrons/DN)
(snoise =         0.) ccdclip: Sensitivity noise (fraction)
(pclip  =       -0.5) pclip: Percentile clipping parameter
```

```
(blank  =         1.) Value if there are no pixels
(mode   =          q)
```

Além de obter um FLAT médio, neste caso, ainda precisamos normalizar o FLAT, pois como foi obtido o FLAT de lâmpada ele tem componentes dependentes do comprimento da lâmpada como *bumps*, *wiggles* e efeitos de cor. Esses efeitos são retirados após a normalização do FLAT médio por ele mesmo e isso é feito pela tarefa RESPONSE com os parâmetros que se seguem abaixo.

```
   TASK = response

calibrat=  flat.fits  Longslit calibration images
normaliz=  flat.fits  Normalization spectrum images
response= nflat.fits  Response function images
(interac=        yes) Fit normalization spectrum interactively?
(thresho=      INDEF) Response threshold
(sample =          *) Sample of points to use in fit
(naverag=          5) Number of points in sample averaging
(functio=    spline3) Fitting function
(order  =          5) Order of fitting function
(low_rej=         3.) Low rejection in sigma of fit
(high_re=         3.) High rejection in sigma of fit
(niterat=          1) Number of rejection iterations
(grow   =         0.) Rejection growing radius
(graphic=   stdgraph) Graphics output device
(cursor =           ) Graphics cursor input
(mode   =          q)
```

Agora só nos resta dividir as imagens de ciência pelo FLAT médio normalizado, e isto é feito, mais uma vez, pelo CCDPROC.

Pronto, correções feitas podemos partir para a extração do espectro.

5. EXTRAÇÃO DO ESPECTRO

A extração dos espectros é feita pela tarefa APALL do pacote NOAO.ONEDSPEC.KPNOSLIT. Essa tarefa também é encontrada em outros pacotes do IRAF, o que muda de um para o outro é a maneira como são feitos os cálculos para a extração do espectro.

Nós iremos usar esta tarefa duas vezes, uma para a redução de um espectro individualmente ou para uma lista de espectros, e outra em modo automático para uma lista de espectros e lâmpadas de calibração com as definições ajustadas pela extração do espectro individual.

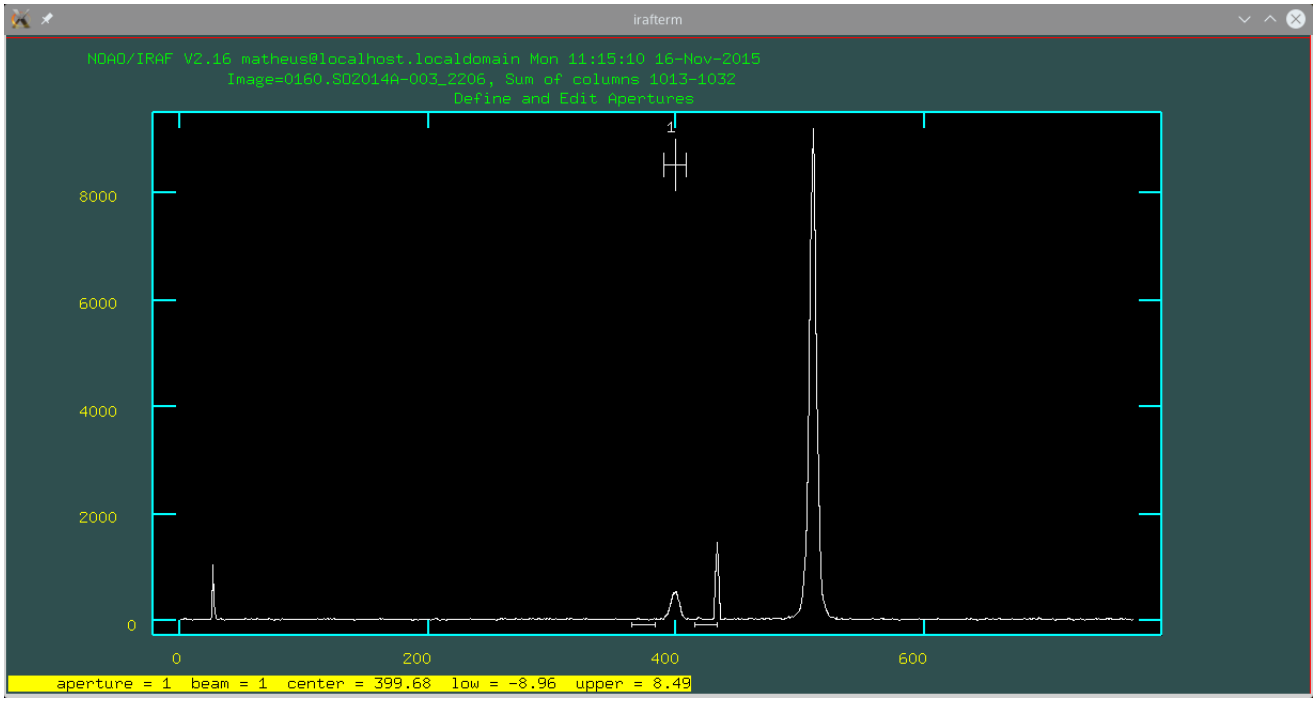
Para a extração do espectro individual e/ou para a lista de espectros utilizamos os parâmetros abaixo.

```
TASK = apall

input   = 0152.SO2014A-003_2206.fits List of input images
(output =          ) List of output spectra
(apertur=          ) Apertures
(format = multispec) Extracted spectra format
(referen=          ) List of aperture reference images
```

```
(profile=          ) List of aperture profile images

(interac=       yes) Run task interactively?
(find   =       yes) Find apertures?
(recente=       yes) Recenter apertures?
(resize =        no) Resize apertures?
(edit   =       yes) Edit apertures?
(trace  =       yes) Trace apertures?
(fittrac=       yes) Fit the traced points interactively?
(extract=       yes) Extract spectra?
(extras =       yes) Extract sky, sigma, etc.?
(review =       yes) Review extractions?

(line   =     INDEF) Dispersion line
(nsum   =        20) Number of dispersion lines to sum or median

                  # DEFAULT APERTURE PARAMETERS

(lower  =      -10.) Lower aperture limit relative to center
(upper  =       10.) Upper aperture limit relative to center
(apidtab=          ) Aperture ID table (optional)

                  # DEFAULT BACKGROUND PARAMETERS

(b_funct= chebyshev) Background function
(b_order=         1) Background function order
(b_sampl= -35:-15,15:35) Background sample regions
(b_naver=      -100) Background average or median
(b_niter=        10) Background rejection iterations
(b_low_r=        3.) Background lower rejection sigma
(b_high_=        2.) Background upper rejection sigma
(b_grow =        4.) Background rejection growing radius

                  # APERTURE CENTERING PARAMETERS

(width  =       10.) Profile centering width
(radius =       10.) Profile centering radius
(thresho=        0.) Detection threshold for profile centering

                  # AUTOMATIC FINDING AND ORDERING PARAMETERS

nfind   =         1  Number of apertures to be found automatically
(minsep =        5.) Minimum separation between spectra
(maxsep =     1000.) Maximum separation between spectra
(order  = increasing) Order of apertures

                   # RECENTERING PARAMETERS

(aprecen=          ) Apertures for recentering calculation
(npeaks =     INDEF) Select brightest peaks
(shift  =       yes) Use average shift instead of recentering?

                   # RESIZING PARAMETERS

(llimit =     INDEF) Lower aperture limit relative to center
(ulimit =     INDEF) Upper aperture limit relative to center
```

```
(ylevel =   0.1) Fraction of peak or intensity for automatic width
(peak   =       yes) Is ylevel a fraction of the peak?
(bkg    =       yes) Subtract background in automatic width?
(r_grow =        0.) Grow limits by this factor
(avglimi=        no) Average limits over all apertures?

                  # TRACING PARAMETERS

(t_nsum =        10) Number of dispersion lines to sum
(t_step =        10) Tracing step
(t_nlost= 3) Number of consecutive times profile is lost before
quitting
(t_funct= legendre) Trace fitting function
(t_order=         3) Trace fitting function order
(t_sampl=         *) Trace sample regions
(t_naver=         1) Trace average or median
(t_niter=         1) Trace rejection iterations
(t_low_r=        3.) Trace lower rejection sigma
(t_high_=        3.) Trace upper rejection sigma
(t_grow =        0.) Trace rejection growing radius

                  # EXTRACTION PARAMETERS

(backgro=       fit) Background to subtract
(skybox =         1) Box car smoothing length for sky
(weights= variance) Extraction weights (none|variance)
(pfit   =    fit2d) Profile fitting type (fit1d|fit2d)
(clean  =      yes) Detect and replace bad pixels?
(saturat=    INDEF) Saturation level
(readnoi=     3.99) Read out noise sigma (photons)
(gain   =     2.06) Photon gain (photons/data number)
(lsigma =       4.) Lower rejection threshold
(usigma =       4.) Upper rejection threshold
(nsubaps=         1) Number of subapertures per aperture
(mode   =         q)
```

Notem que para a extração da lista basta alterar o nome da imagem para o nome da lista precedido de um @. O APALL é uma tarefa interativa onde o usuário pode definir previamente (nos parâmetros) o tamanho da abertura e outras definições, e após rodar caso não seja encontrado a fenda de interesse o usuário pode redefinir interativamente a fenda. Na figura abaixo podemos ver a primeira janela interativa que é aberta.



Para interagir com esta janela usamos alguns comandos específicos. Caso queira maiores informações sobre os comandos use ? na janela. Abaixo listarei os comandos que utilizamos para extrair o espectro.

d – exclui a fenda pré definida

w – ativa o modo da janela, onde podemos dar zoom (e duas vezes em diferentes regiões) e retornar para a imagem completa (a), para mais comandos utilize ?

m – define a nova abertura próxima do ponteiro

u – define a parte superior da abertura

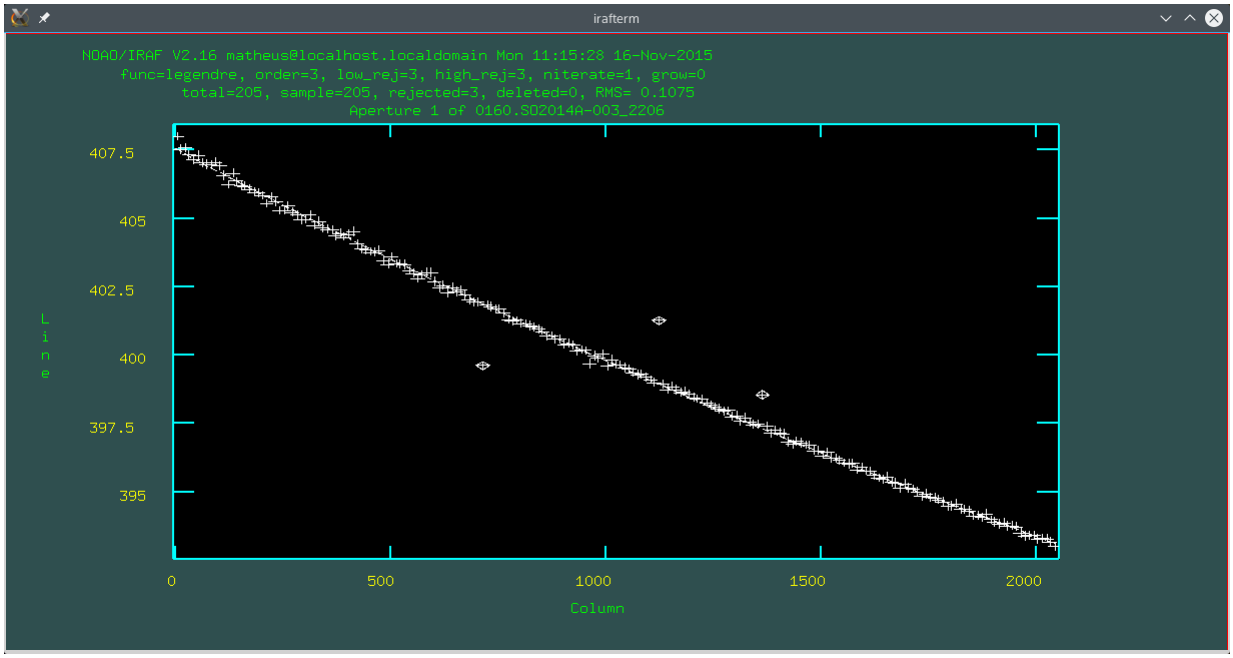
l – define a parte inferior da abertura

b – abre a janela de céu

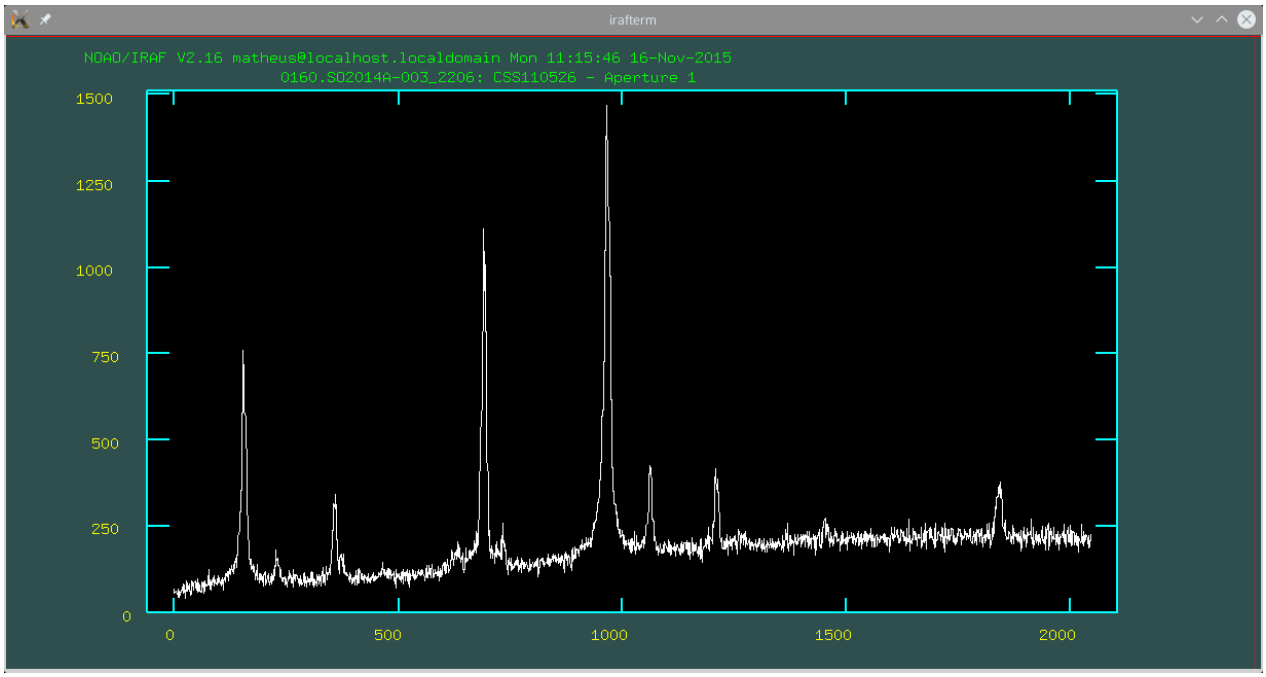
z – exclui a janela de céu próxima

s – define nova janela de céu, basta clicar duas vezes em diferente lugares.

Agora que definimos a abertura e a região do céu usamos q para seguir adiante. Notem que ele traça a abertura (figura abaixo) de acordo com a função especificada, neste caso uma legendre de ordem 3. Caso o RMS esteja alto e o traço não esteja bom podemos alterar a ordem da função ou a função com os comandos :o (numero) e :f (nome_da_função), para maiores informações basta digitar ?.



Quando estivermos satisfeitos basta aperta q e seguir as informações descritas na janela (basta confirmar tudo com yes) caso já tenha extraído a última pergunta será se deseja sobrescrever, e por fim ele mostra uma visualização do espectro extraído.



Extraímos o primeiro espectro, agora temos que repetir para os demais e para as lâmpadas. Mas calma aí, não será tão difícil, como foi dito anteriormente o APALL faz a extração automática depois que temos o primeiro, para isso basta alterar os parâmetros do APALL como descritos abaixo.

```
TASK = apall

input    =     @todos  List of input images
(output =           ) List of output spectra
```

```
(apertur=          ) Apertures
(format = multispec) Extracted spectra format
(referen=  0152.SO2014A-003_2206)  List  of  aperture  reference
images
(profile=          ) List of aperture profile images

(interac=       yes) Run task interactively?
(find   =        no) Find apertures?
(recente=        no) Recenter apertures?
(resize =        no) Resize apertures?
(edit   =        no) Edit apertures?
(trace  =        no) Trace apertures?
(fittrac=        no) Fit the traced points interactively?
(extract=       yes) Extract spectra?
(extras =        no) Extract sky, sigma, etc.?
(review =       yes) Review extractions?

(line   =     INDEF) Dispersion line
(nsum   =        20) Number of dispersion lines to sum or median

                # DEFAULT APERTURE PARAMETERS

(lower  =      -10.) Lower aperture limit relative to center
(upper  =       10.) Upper aperture limit relative to center
(apidtab=          ) Aperture ID table (optional)

                # DEFAULT BACKGROUND PARAMETERS

(b_funct= chebyshev) Background function
(b_order=         1) Background function order
(b_sampl= -35:-15,15:35) Background sample regions
(b_naver=      -100) Background average or median
(b_niter=        10) Background rejection iterations
(b_low_r=        3.) Background lower rejection sigma
(b_high_=        2.) Background upper rejection sigma
(b_grow =        4.) Background rejection growing radius

                # APERTURE CENTERING PARAMETERS

(width  =       10.) Profile centering width
(radius =       10.) Profile centering radius
(thresho=        0.) Detection threshold for profile centering

                # AUTOMATIC FINDING AND ORDERING PARAMETERS

nfind   =      1  Number of apertures to be found automatically
(minsep =     5.) Minimum separation between spectra
(maxsep =  1000.) Maximum separation between spectra
(order  = increasing) Order of apertures

                # RECENTERING PARAMETERS

(aprecen=       ) Apertures for recentering calculation
(npeaks =  INDEF) Select brightest peaks
(shift  =    yes) Use average shift instead of recentering?
```

```
                # RESIZING PARAMETERS

(llimit =  INDEF) Lower aperture limit relative to center
(ulimit =  INDEF) Upper aperture limit relative to center
(ylevel =   0.1) Fraction of peak or intensity for automatic width
(peak   =     yes) Is ylevel a fraction of the peak?
(bkg    =     yes) Subtract background in automatic width?
(r_grow =      0.) Grow limits by this factor
(avglimi=      no) Average limits over all apertures?

                # TRACING PARAMETERS

(t_nsum =      10) Number of dispersion lines to sum
(t_step =      10) Tracing step
(t_nlost= 3) Number of consecutive times profile is lost before
quitting
(t_funct= legendre) Trace fitting function
(t_order=         3) Trace fitting function order
(t_sampl=         *) Trace sample regions
(t_naver=         1) Trace average or median
(t_niter=         1) Trace rejection iterations
(t_low_r=        3.) Trace lower rejection sigma
(t_high_=        3.) Trace upper rejection sigma
(t_grow =        0.) Trace rejection growing radius

                  # EXTRACTION PARAMETERS

(backgro=      none) Background to subtract
(skybox =         1) Box car smoothing length for sky
(weights= variance) Extraction weights (none|variance)
(pfit   =     fit2d) Profile fitting type (fit1d|fit2d)
(clean  =        no) Detect and replace bad pixels?
(saturat=     INDEF) Saturation level
(readnoi=      3.99) Read out noise sigma (photons)
(gain   =      2.06) Photon gain (photons/data number)
(lsigma =        4.) Lower rejection threshold
(usigma =        4.) Upper rejection threshold
(nsubaps=         1) Number of subapertures per aperture
(mode   =         q)
```

Notem que o campo mais importante é o reference, é nele que você irá indicar qual o espectro de referência para os demais. Depois de indicado o APALL irá buscar o espectro de referência dentro da pasta database que ele mesmo criou, não se preocupe. Caso tenha dúvidas de como colocar o nome do espectro de referência é só dar uma olhada na pasta.

Caso queira ver os espectros extraídos depois podemos usar a tarefa SPLOT da seguinte forma

splot nome_do_espectro.ms.fits

Notem que o nome do espectro agora tem a terminação .ms isso é o padrão do APALL, para trabalharmos com os espectros daqui pra frente precisaremos utilizar as imagens com esta terminação.

Pronto, espectros extraídos, salvos, agora só nos resta fazer as calibrações.

6. CALIBRAÇÃO EM COMPRIMENTO DE ONDA

A calibração em comprimento de onda, na minha opinião, é a mais importante para a espectroscopia. Pois é a partir dela que podemos identificar as linhas do nosso espectro para determinados elementos.

Para fazer a calibração em comprimento de onda, precisamos usar os espectros das lâmpadas extraídos (com a terminação .ms). Para este tutorial utilizamos a lâmpada de CuAr instalada no SOAR.

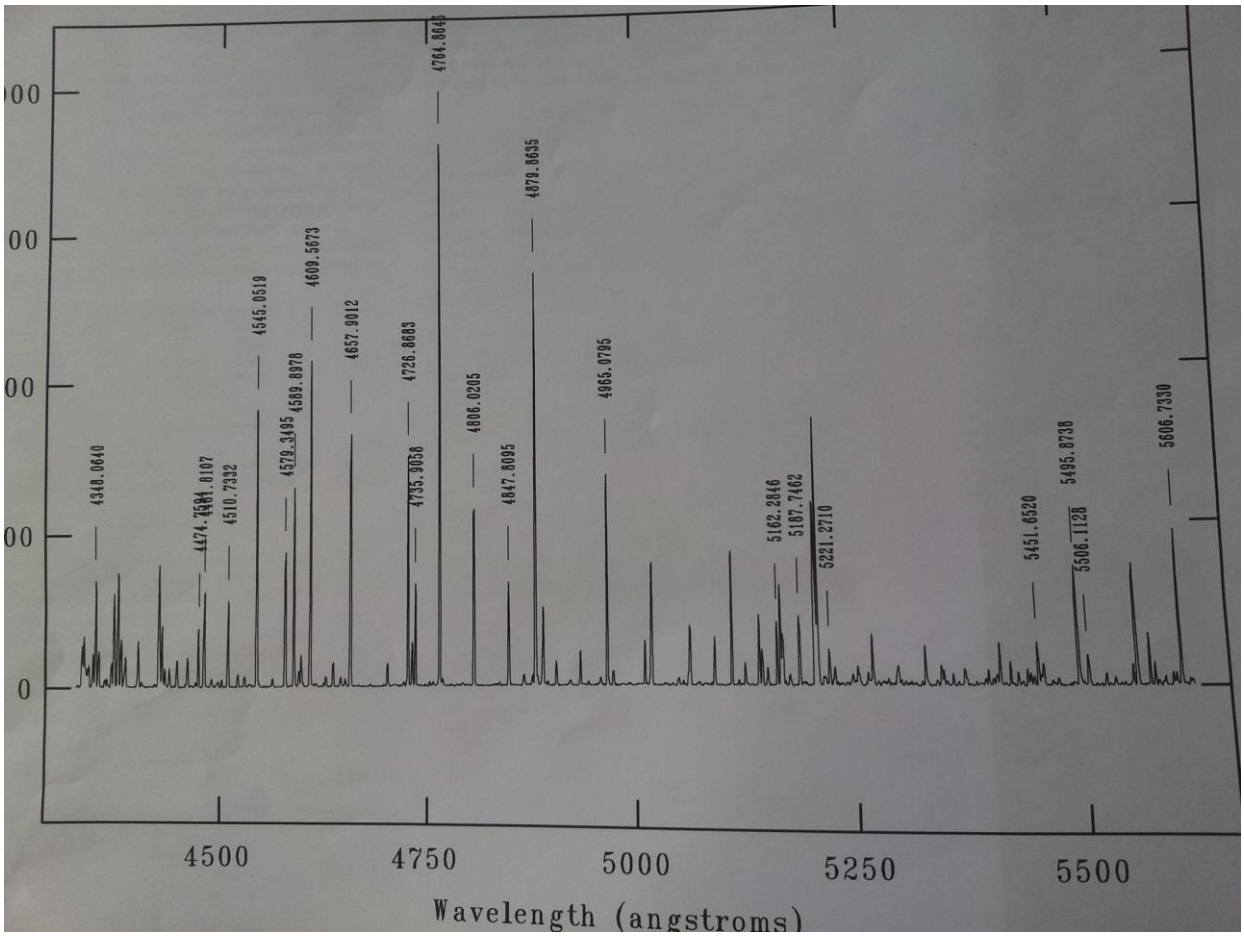
A tarefa que nos permite identificar os comprimentos de onda é a IDENTIFY, ela utiliza uma tabela com valores dos comprimentos de onda e o elemento identificado naquele comprimento de onda, o IRAF possui um banco de dados para estas lâmpadas que pode ser encontrado em:

cd linelists$

Caso o observatório possua uma tabela com a identificação dos comprimentos de lâmpada é melhor usá-la do que a genérica do IRAF. Este é o nosso caso, o SOAR disponibiliza estes dados para nós, então temos que o declara-lo no IDENTIFY como podemos ver nos parâmetros abaixo:

```
   TASK = identify

images     =   0165.SO2014A-003_2206.ms.fits  Images  containing
features to be identified
(section= middle line) Section to apply to two dimensional images
(databas= database) Database in which to record feature data
(coordli= linelists$goodman_cuhear.dat) User coordinate list
(units  =     ) Coordinate units
(nsum   =    10) Number of lines/columns/bands to sum in 2D images
(match  =  10.) Coordinate list matching limit
(maxfeat=   50)  Maximum  number  of  features  for  automatic
identification
(zwidth =  100.) Zoom graph width in user units
(ftype  = emission) Feature type
(fwidth =    4.) Feature width in pixels
(cradius=    5.) Centering radius in pixels
(thresho=    0.) Feature threshold for centering
(minsep =    2.) Minimum pixel separation
(functio= legendre) Coordinate function
(order  =     4) Order of coordinate function
(sample =     *) Coordinate sample regions
(niterat=        1) Rejection iterations
(low_rej=       3.) Lower rejection sigma
(high_re=       3.) Upper rejection sigma
(grow   =       0.) Rejection growing radius
(autowri=      yes) Automatically write to database
(graphic= stdgraph) Graphics output device
(cursor =         ) Graphics cursor input
crval   =           Approximate coordinate (at reference pixel)
cdelt   =           Approximate dispersion
(aidpars=         ) Automatic identification algorithm parameters
(mode   =       ql)
```

Para utilizar o IDENTIFY nós temos que ter em mãos um espectro da lâmpada já identificado como na figura abaixo



Agora no IDENTIFY por comparação temos que achar as linhas que correspondem as linhas já identificadas. Quando acharmos nos marcamos a linha apertando m e digitamos o valor daquele comprimento de onda, depois de identificar umas 2 ou 3 basta apertar f, isso faz com que o IDENTIFY gere uma função para esta lâmpada e assim fique mais fácil identificar as demais linhas. Para retornar apertamos q e continuamos a identificar as linhas (podemos ver estes processos nas figuras abaixo). Quando tivermos certeza das linhas identificadas basta apertar l que todas as demais linhas serão automaticamente identificadas. Pronto, criado o arquivo das linhas.

Agora precisamos definir qual lâmpada será utilizada para calibrar o espectro, para isso podemos simplesmente editar o header das imagens com a lâmpada de referência ou utilizar a tarefa REFSPEC.

```
   TASK = refspectra

input   =     @espec  List of input spectra
(referen=   0155.SO2014A-003_2206.ms.fits)  List  of  reference
spectra
(apertur=           ) Input aperture selection list
(refaps =           ) Reference aperture selection list
(ignorea=        yes) Ignore input and reference apertures?
(select =     interp) Selection method for reference spectra
```

```
(sort   =         jd) Sort key
(group  =        ljd) Group key
(time   =         no) Is sort key a time?
(timewra=        17.) Time wrap point for time sorting
(overrid=         no) Override previous assignments?
(confirm=        yes) Confirm reference spectrum assignments?
(assign = yes) Assign the reference spectra to the input spectrum?
(logfile= STDOUT,logfile) List of logfiles
(verbose=         no) Verbose log output?
answer  =        yes  Accept assignment?
(mode   =         ql)
```

Caso decida editar o header temos que incluir o seguinte termo REFSPEC1=”espectro_da_lampada”, para incluir podemos utilizar o seguinte comando:

hedit nome_do_espectro.ms.fits REFSPEC1=”espectro_da_lampada” add+ ver-show+

Feito isso basta utilizar agora, por fim, a tarefa DISPCOR com os parâmetros abaixo que temos nossos espectros calibrados em comprimento de onda.

```
   TASK = dispcor

input   =      @espec  List of input spectra
output  =   d//@espec  List of output spectra
(lineari=         yes) Linearize (interpolate) spectra?
(databas=    database) Dispersion solution database
(table  =            ) Wavelength table for apertures
(w1     =       INDEF) Starting wavelength
(w2     =       INDEF) Ending wavelength
(dw     =       INDEF) Wavelength interval per pixel
(nw     =       INDEF) Number of output pixels
(log    =          no) Logarithmic wavelength scale?
(flux   =         yes) Conserve total flux?
(blank  =          0.) Output value of points not in input
(samedis=          no) Same dispersion in all apertures?
(global =          no) Apply global defaults?
(ignorea=          no) Ignore apertures?
(confirm=          no) Confirm dispersion coordinates?
(listonl=          no) List the dispersion coordinates only?
(verbose=         yes) Print linear dispersion assignments?
(logfile=            ) Log file
(mode   =          ql)
```

7. CALIBRAÇÃO EM FLUXO

Caso na observação foi feita uma estrela espectrofotométrica padrão podemos calibrar o fluxo do nosso espectro com ela. No nosso caso foi feita a espectroscopia da padrão LTT6248, e para a calibração do fluxo utilizamos as seguintes tarefas SETAIRMASS, STANDARD, SENSFUNC e finalmente o CALIBRATE, os parâmetros estão abaixo:

```
   TASK = setairmass

images  =     @spec  Input images
(observa=      soar) Observatory for images
```

```
(intype = beginning) Input keyword time stamp
(outtype= effective) Output airmass time stamp

(ra     =        RA) Right acsension keyword (hours)
(dec    =       DEC) Declination keyword (degrees)
(equinox=   EQUINOX) Equinox keyword (years)
(st     =       LST) Local siderial time keyword (hours)
(ut     =        UT) Universal time keyword (hours)
(date   =  DATE-OBS) Observation date keyword
(exposur=   EXPTIME) Exposure time keyword (seconds)
(airmass=   AIRMASS) Airmass keyword (output)
(utmiddl=  UTMIDDLE) Mid-observation UT keyword (output)
(scale  =      750.) The atmospheric scale height
(show   =       yes) Print the airmasses and mid-UT?
(update =       yes) Update the image header?
(overrid=       yes) Override previous assignments?
(mode   =         q)

   TASK = standard

input   = 0154.S02014A-003_2206.ms.fits Input image file root name
output  =        std  Output flux file (used by SENSFUNC)
(samesta=        yes) Same star in all apertures?
(beam_sw=         no) Beam switch spectra?
(apertur=           ) Aperture selection list
(bandwid=      INDEF) Bandpass widths
(bandsep=      INDEF) Bandpass separation
(fnuzero= 3.6800000000000E-20) Absolute flux zero point
(extinct= onedstds$ctioextinct.dat) Extinction file
(caldir = onedstds$ctionewcal/) Directory containing calibration
data
(observa=       soar) Observatory for data
(interac=        yes) Graphic interaction to define new bandpasses
(graphic=   stdgraph) Graphics output device
(cursor =           ) Graphics cursor input
star_nam=     LTT6248 Star name in calibration list
airmass =             Airmass
exptime =             Exposure time (seconds)
mag     =             Magnitude of star
magband =             Magnitude type
teff    =             Effective temperature or spectral type
answer  =         no  (no|yes|NO|YES|NO!|YES!)
(mode   =         ql)

   TASK = sensfunc

standard=      std  Input standard star data file (from STANDARD)
sensitiv=     sens  Output root sensitivity function imagename
(apertur=         ) Aperture selection list
(ignorea= yes) Ignore apertures and make one sensitivity function?
(logfile=  logfile) Output log for statistics information
(extinct= onedstds$ctioextinct.dat) Extinction file
(newexti=         ) Output revised extinction file
(observa=     soar) Observatory of data
(functio=  spline3) Fitting function
(order  =        9) Order of fit
```

```
(interac=      yes) Determine sensitivity function interactively?
(graphs =       sr) Graphs per frame
(marks  = plus cross box) Data mark types (marks deleted added)
(colors =  2 1 3 4) Colors (lines marks deleted added)
(cursor =         ) Graphics cursor input
(device = stdgraph) Graphics output device
answer  =      yes (no|yes|NO|YES)
(mode   =       ql)

   TASK = calibrate

input   =     @spec  Input spectra to calibrate
output  =  c//@spec  Output calibrated spectra
(extinct=       yes) Apply extinction correction?
(flux   =       yes) Apply flux calibration?
(extinct= onedstds$ctioextinct.dat) Extinction file
(observa=      soar) Observatory of observation
(ignorea=      yes) Ignore aperture numbers in flux calibration?
(sensiti=     sens) Image root name for sensitivity spectra
(fnu    =       no) Create spectra having units of FNU?
airmass =           Airmass
exptime =           Exposure time (seconds)
(mode   =      ql)
```

Pronto, agora temos nosso espectro finalizado e podemos fazer ciência com ele.

8. REFERÊNCIAS

Este guia foi baseado nas apostilas de Philip Massey e colaboradores que pode ser encontrado em ftp://iraf.noao.edu/ftp/docs/ . As apostilas abaixo foram usadas como referência:

MASSEY, Philip. A User's Guide to CCD Reduction with IRAF, 15 de fevereiro de 1997. Nome encontrado no ftp acima: ccduser3

MASSEY, Philip, VALDES, Frank, BARNES, Jeannette. A User's Guide to Reducing Slit Spectra with IRAF, 15 de abril de 1992. Nome encontrado no ftp acima: spect