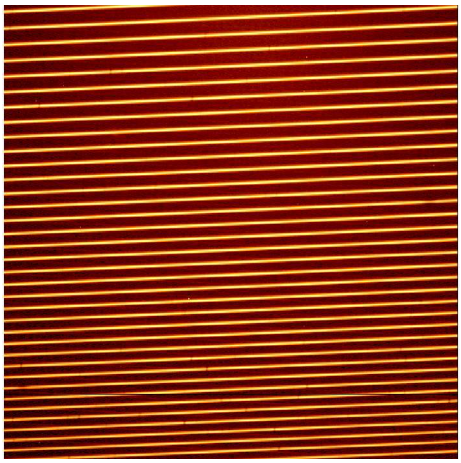
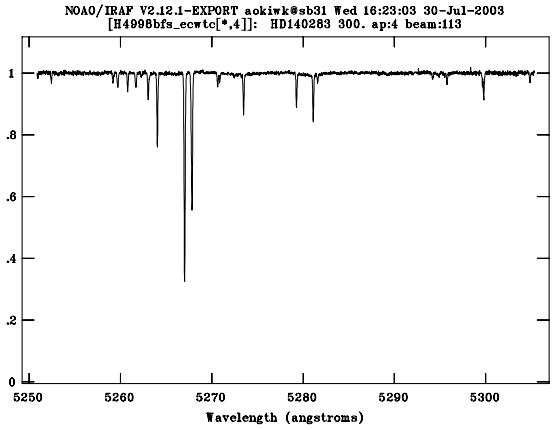
すばる/HDSデータ解析講習会資料

IRAFを用いたスペクトル解析

2008年5月改訂版





青木和光
国立天文台

はじめに

天文学において、可視光の分光観測は天体の速度構造や物理状態を探る手段として、標準的に用いられます。特に高分散分光観測は、恒星物理学においては、星の大気構造（表面の構造）の理解に大きく寄与し、ひいては星の内部構造や進化についても情報をもたらします。また、恒星や QSO を光源とした星間物質や銀河間物質の高分散分光観測も進められ、宇宙の物質進化の理解が進んできています。

この講習会では、すばる望遠鏡高分散分光器 (HDS) で取得された CCD データを、生データの段階から処理し、実際にスペクトルを得る作業を行います。そして、得られたスペクトルからどんな情報が得られるのか考察します。用いるのはいずれも恒星の観測データですが、この作業はさまざまな天体の解析に応用可能です。また、岡山天体物理観測所の HIDES をはじめ、エシェル分光器によるデータの整約一般にも応用できます。

データの処理は、IRAF (Image Reduction and Analysis Facility) [1] という汎用のデータ処理ソフトウエアパッケージを用いて行います。

なお、もう少し詳しい手引（「IRAF によるデータ整約の手引き」）は以下から取得可能です。

`http://optik2.mtk.nao.ac.jp/HDS/index_J.html`

また、IRAF のサイトからにもマニュアルがいくつか用意されていますので、必要に応じて取得してください。

エシェルスペクトルデータの整約に関する疑問やコメントがありましたら、下記までお寄せください。

2005 年 3 月 11 日

青木和光
〒 181-8588 三鷹市大沢 2-21-1 国立天文台
TEL: 0422-34-3531　FAX:0422-34-3545
E-mail: aoki.wako@nao.ac.jp

2007 年 11 月の改訂：このテキストはすばる望遠鏡のデータ解析講習や東大学生実習のテキストとして使ってきましたが、そこで見つかった誤植等を直すほか、IRAF 利用のコツや陥りやすい間違いについて説明を加えることにしました。また、学生実習で用いているぐんま天文台の分光器では、スリット像が CCD のピクセル並びに揃っていないという特徴があるので、それへの対応を加えました。

2008 年 5 月の改訂： いくつかのタスク (apnormalize, ecreidentify, refspectra) およびそのパラメータ設定の説明を補充しました。

[1] 詳しくは http://iraf.noao.edu/ 参照

目 次

1　エシェル分光器で得られるスペクトルデータ

撮像データが、検出器上に写された天空の二次元画像であるのに対し、スペクトルデータの場合には、天体の光は分光器をとおすことにより波長方向に分解され、検出器の一方向に記録されます。つまり、波長分散される方向（ここで扱うスリット分光データの場合は、スリット幅の方向）の空間情報はこの場合捨てられます。それと垂直な方向（スリット長方向）の空間情報はもう一方の軸に記録されます。今回あつかうような星のデータの場合、星像の断面が（波長ごとに）記録されることになります。

最近は、高い波長分解能のスペクトルを得るためには、グレーティングの高い干渉次数を使う「エシェル (Echelle) 分光器」が用いられるようになってきています（用いるグレーティングはエシェルグレーティング、または単にエシェルと呼ばれます）。HDS や HIDES もエシェル分光器です。このようなグレーティングの使いかたをすると、異なる干渉次数に対応するスペクトルが重なって得られることになりますので、これを分離するために、もうひとつ別の（分散度の低い）グレーティングを用いて、エシェルによる波長分散とは垂直方向に光を分散します。これにより、スペクトルは、検出器上に折り畳んで記録されることになり、2 次元検出器を有効に利用できます。

HDS で得られたスペクトルデータの例を図 1 に示します。横軸がエシェルによって波長分散される方向です（左が短波長）。何本も移っているスペクトルは、それぞれエシェルの干渉次数に対応したスペクトルで、下にいくほど高い干渉次数（短波長）になっています。

個々のスペクトルについては、縦方向が、スリット長方向にあたりますので、空間情報が記録されることになります。このデータの縦方向の断面図を図 2 に示します。個々のスペクトルについては、スリット長方向の光の分布、すなわち星像の断面を見ていることになります。星の観測の場合、空間的な広がりは意味がありませんので、星像方向のデータを足し合わせることで、より精度の高いデータを得ます（一次元スペクトルの抽出）。

2　作業の流れ

データの処理は、まず、CCD データの特性を補正する作業からはじまります。データのゼロ点補正を行うバイアス補正、光が当たっていない場合にもカウントがのってしまう現象（暗電流、もしくはダーク）の補正、CCD のピクセルごとの感度ムラの補正（フラットフィールディング）を行います。これらは二次元画像を扱う作業で、基本的には撮像データの処理でも行う作業です。

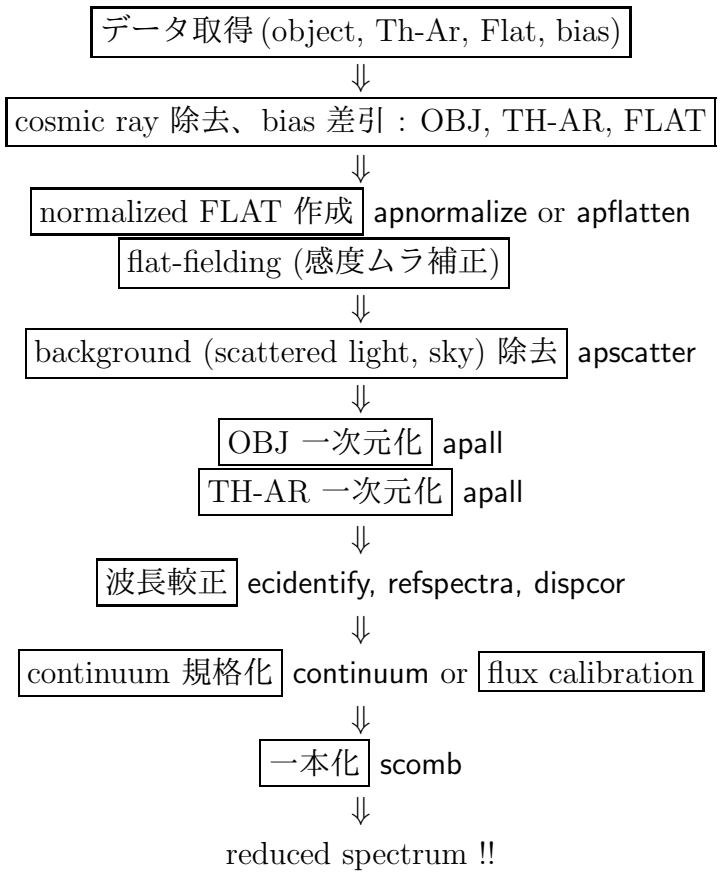
図 2 のデータではよく見えないかもしれませんが、スペクトルどうしの間も、カウントは完全にゼロにはなっていません。これは分光器内の散乱光などによるもので、これを差し引く作業が必要です。

これらの補正を行った上で、スペクトルの写っている位置を測定し、トレースすることにより、一次元のスペクトルを得ます。

得られた一次元スペクトルは、横軸が CCD のピクセル数、縦軸が CCD のカウントになっています。この横軸を実際の波長に直す作業が必要です（波長較正）。縦軸については、ここでは絶対値を無視して、星の光の連続光（コンティニューム）成分で規格化します。

こうして得られたスペクトルを用いて、その後の解析を行います。

以下にデータリダクションの概要を示します。



3　データの準備

まず、処理するデータの準備から行いましょう。この講習は、すばるのデータアーカイブシステム STARS からデータを取得するところから始めます (別資料参照)。

天文データは、多くの場合、FITS とよばれる形式で保存されるようになっています。FITS データの特徴等についてはここでは触れませんが、データについての説明を記述したヘッダ部と、データ本体が一つのファイルにまとめられています。そして、IRAF は FITS 形式のデータの読み書きが可能ですので、とくに気を使わずにデータ処理を進めることができます。

まずは、解析を行う天体のデータを選びます。複数回観測している場合には、すべて選んでおきましょう。また、その前後でもっとも近い時刻に、同じ波長設定とってある比較光源のデータ (comparison) とフラットデータ (flat) を選びましょう。フラットデータも複数枚あると思いますが、すべて選んでおきます。

それから、バイアスデータを選びます。バイアスデータは、観測開始時と終了時にまとめて数枚とっておくことが多いですが、観測中にとってある場合もあります。できれば天体の観測に近い時刻のものを数枚選びましょう。

以上のリストをつくり、データを自分のディレクトリにコピーします。

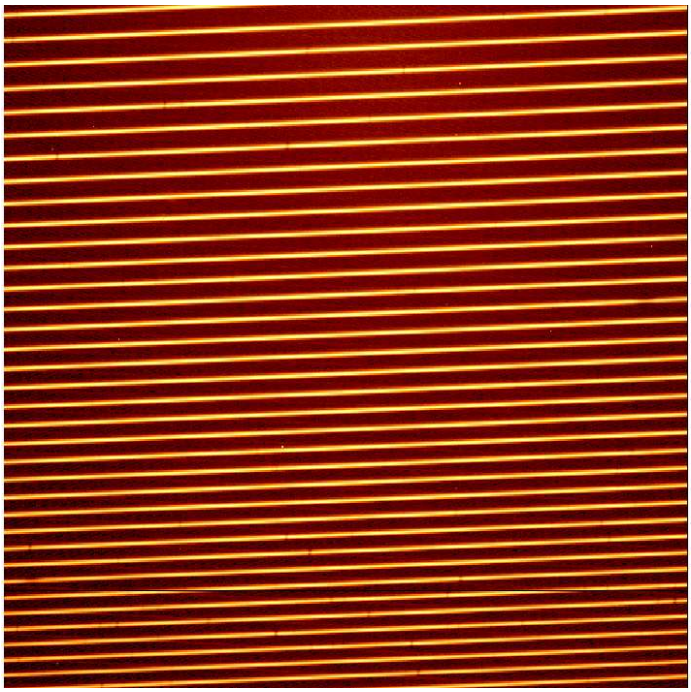


図 1: 天体のデータの二次元画像。横が波長分散方向、縦がスリット長方向。詳しくは本文参照。

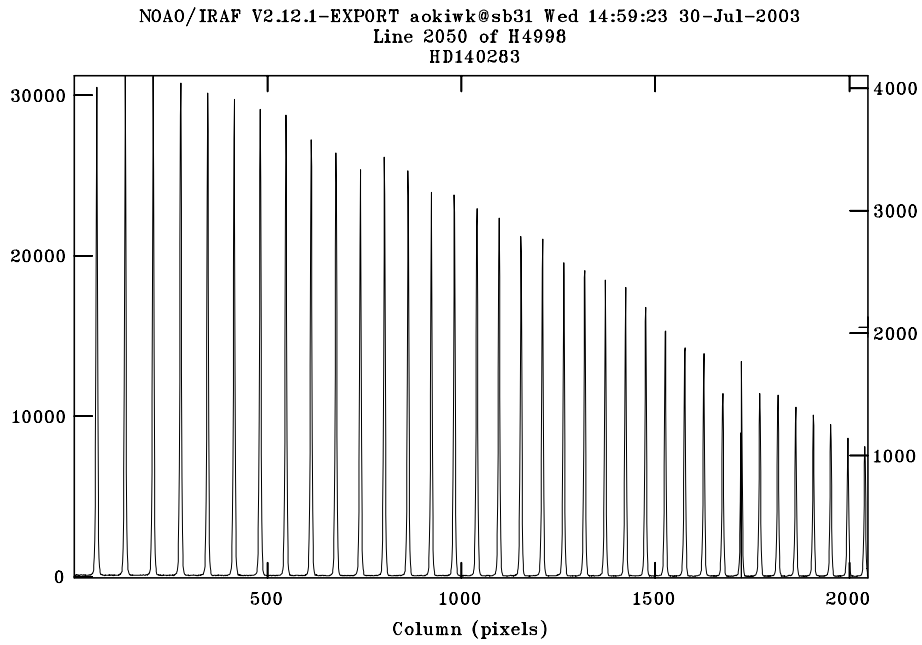


図 2: 天体データの断面（スリット長方向）

4 IRAF の準備

作業をはじめるにあたって、IRAF の準備を行います。

まずは、計算機のウインドウシステムを立ち上げたうえで、xgterm もしくは xterm を立ち上げます (xgterm がおすすめ)。そして、自分のホームディレクトリで、' mkiraf' を実行します。すると、「初期化するか？」「ターミナルのタイプは？」と聞いてくるので、以下のように答えます。

```
r22{aokiwk}2: mkiraf
Initialize uparm? (y|n): y
-- initializing uparm
Terminal types: xgterm,xterm,gterm,vt640,vt100,etc.
Enter terminal type: xgterm
A new LOGIN.CL file has been created in the current directory.
You may wish to review and edit this file to change the defaults.
```

これで準備 OK です。ホームディレクトリに login.cl というファイルができています。このファイルを書き換えることによって IRAF に関する設定ができますが、今は特に変更は必要ありません。

必須ではありませんが、デフォルトで取り扱うイメージタイプを FITS にしておくと便利です。この場合、

```
#set     imtype          = "imh"
```

の行を、

```
set     imtype          = "fits"
```

にしておきます。

IRAF を起動するには、ホームディレクトリで、'cl' を実行します。

```
r22{aokiwk}2: cl
```

すると、以下のように表示されて、IRAF が使えるようになります。

```
# LOGIN.CL -- User login file for the IRAF command language.
    NOAO Sun/IRAF Revision 2.11.3 Sat Sep  9 23:18:55 MST 2000
    This is the EXPORT version of Sun/IRAF V2.11 for SunOS 4 and Solaris 2.7

    Welcome to IRAF.  To list the available commands, type ? or ??.  To get
    detailed information about a command, type `help command'.  To run a
    command or load a package, type its name.  Type `bye' to exit a
    package, or `logout' to get out of the CL.  Type `news' to find out
    what is new in the version of the system you are using.  The following
    commands or packages are currently defined:

      apropos     images.     noao.       proto.      stsdas.     utilities.
      dataio.     language.   obsolete.   softools.   system.
      dbms.       lists.      plot.       spiral.     tables.
cl>
```

気をつけなければならないのが、IRAF の起動は、login.cl のあるディレクトリ（ここではホームディレクトリ）で行う必要があることです。それ以外のディレクトリでも IRAF は立ち上がりますが、login.cl にある必要な情報がとれないことにより、非常に作業が不便です。必ず login.cl をつくったホームディレクトリで cl を実行しましょう。

作業するディレクトリに 'cd' コマンドで移動して、作業にかかります。

IRAF を用いた作業は、IRAF のコマンド（タスクと呼びます）を用いて個々に行います。タスクは、種類ごとにパッケージに分けられており、それぞれのパッケージを開いて（というよりパッケージに移って）実行します。たとえば、一次元スペクトルの抽出を行う apall というタスクは、 imred というパッケージのなかの、echelle というパッケージにあります。それぞれのパッケージに移るには、パッケージ名を実行すれば OK です。[2] したがって、apall を実行するには、

```
cl> imred
      argus.      crutil.     echelle.    iids.       kpnocoude.  specred.
      bias.       ctioslit.   generic.    irred.      kpnoslit.   vtel.
      ccdred.     dtoi.       hydra.      irs.        quadred.
im> ec
      apall         aprecenter    demos         refspectra    sflip
      apdefault@    apresize      deredden      sapertures    slist
      apedit        apscatter     dispcor       sarith        specplot
      apfind        apsum         doecslit      scombine      specshift
      apfit         aptrace       dofoe         scopy         splot
      apflatten     bplot         dopcor        sensfunc      standard
      apmask        calibrate     ecidentify    setairmass
      apnormalize   continuum     ecreidentify  setjd
ec>
```

という具合に、echelle に移ります。もどるには、'bye' を実行します。各タスクがどのパッケージに含まれるのかは、ヘルプを見ればわかります。ヘルプは、'help タスク名' で見られます：

```
ec> help apall
```

と入力すると、以下のように apsum についての説明が表示されます。パッケージ名が、noao.twodspec.apextract と示されています。ヘルプには有用な情報が書かれていますので、必要に応じてどんどん使いましょう。

```
APALL (Sep96)            noao.twodspec.apextract           APALL (Sep96)

NAME
    apall -- Extract one dimensional sums across the apertures


USAGE
    apall input
```

[2] パッケージ名、タスクとも、全部打ち込む必要はありません。'echelle' は、'ec' までで十分同定できますので、IRAF は補完して実行してくれます。

```
PARAMETERS

......
```

IRAF のタスクを実行する際には、入力ファイル、出力ファイルのほか、多数のパラメータを与える必要がある場合があります。そのパラメータの変更は、`eparam` タスク名' と入力すると、パラメータ設定の画面が出てきますので、そこで書き換えを行います。具体例は、次の節で示します。

5 HDS データの構造とオーバースキャン領域の処理

5.1 データの特徴

- フレーム ID

  HDS データには通し番号 (フレーム ID) がつけられています。HDS では二つの CCD がありますが、それぞれに対して FITS ファイルが作られるため、一回の露出に対して二つのフレーム ID が割り当てられます。フレーム ID は、'HDSA' につづく 8 桁の通し番号です)。番号は遡ることはなく、データ取得を途中でキャンセルするなどした場合には欠番になります。ファイル名は、これに '.fits' をつけたもの (HDSA00002480.fits 等) になっています。

- FITS データの特徴

  取得される FITS データには、通常のヘッダ部、データ部に加え、アスキー拡張テーブル (これもヘッダ部、データ部に分けられる) が添付されます。テーブルには、取得されたスペクトルのフォーマット (回折次数、波長、CCD 上での位置) が記録されます (いずれもグレーティング等の設定からの計算値)。

5.2 データ形式の特徴

データ部は、分散方向 4100 ピクセル、スリット方向 2048 ピクセルのデータ (ビンニング無しの場合) に加えて、オーバースキャン領域がつけられています。オーバースキャンとは、光のあたった部分のデータ読みだしに加えて余分に読みだしを行うことで、そのデータを取得した際のバイアスレベルを記録したものです。

データの読みだしは、それぞれの CCD について 2ヶ所ずつで読みだしているため、データとしては 4100×1024 ピクセルがひとつのユニットとなります。これにそれぞれオーバースキャン領域が 50 列 (4100×50 ピクセル) つけられます。これにより、図 3 のようなデータが得られることになります。なお、以上ではビンニング無しの場合のデータ形式について説明しましたが、ビンニングを行った場合でも、オーバースキャン領域は 50 列ずつつけられます。

バイアスレベルはある程度変動するため、データ整約の際には、オーバースキャン領域のデータを用いて変動分を補正することがのぞましいといえます。整約の行い方については、次節で説明します。

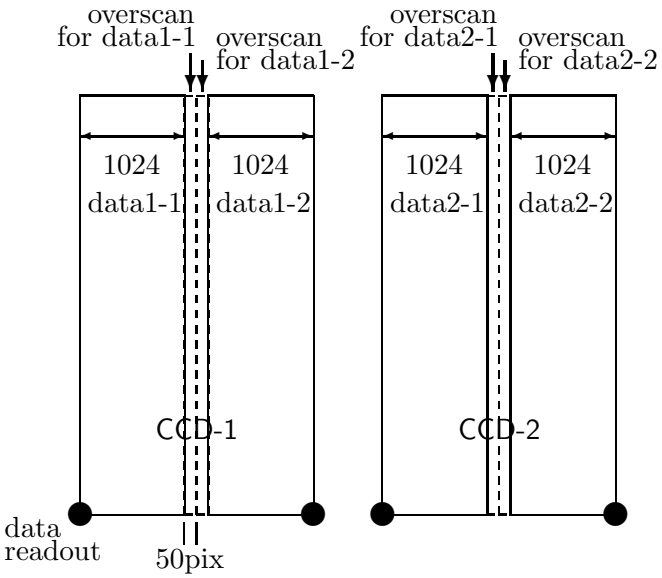


図 3: CCD データ形式の概念図。縦がスペクトル分散方向、横がスリット方向

5.3 HDS データに特有な処理

1. アスキー拡張テーブルの処理

   IRAF によってデータを処理する場合、アスキー拡張テーブルがついていることにより、FITS ファイル名を指定しただけでは直接には処理できません。これは FITS ファイルを読み込むタスク rfits によって処理可能なファイルにできます。

   (例) rfits *input.fits* 0 *output.fits*

   また、ファイル名に [0] をつければ (*HDSA00000001.fits[0]* 等) 直接処理することができます。

2. オーバースキャン領域の処理

   5.2 節で説明したように、HDS のデータにはオーバースキャン領域が付けられています。これは以下のような手順によって有効に処理できます。なお、バイアスレベルの変動が無視できると考えるならば、オーバースキャン領域を切り捨ててしまうこともできます。

   - 各読みだし口に対応するオーバースキャン領域 (50×4100pix) の平均をとる。
   - 各読みだし口ごとに、データから、上で計算したオーバースキャン領域の平均値を差し引く。
   - 各読みだし口に対応するデータごとに、それぞれのゲインをかける。ゲインの値は表 2 のとおり。
   - データ領域だけ切りだし、各 CCD ごとにひとつのファイルになるようにくつなげる (オーバースキャン領域のない 2048×4100pix のデータになる)。

以下に、読みだし口ごとの CCD ゲインの値を示します。なお、この値は、FITS ヘッダにも記載されています (長波長側： `H_GAIN1`、短波長側： `H_GAIN2`)。

表 1: CCD ゲイン

読みだし口	ゲイン (e^-/ADU)
CCD1, 左側 (長波長側)	1.628
CCD1, 右側 (短波長側)	1.615
CCD2, 左側 (長波長側)	1.782
CCD2, 右側 (短波長側)	1.665

この手順で処理を行う IRAF のスクリプト overscan.cl が、以下の URL から取得できます:

`http://optik2.mtk.nao.ac.jp/HDS/index_J.html`

使用方法としては、

```
cl>task overscan=overscan.cl
```

とタスクの定義を行ったうえで、以下のように実行します。

`cl>overscan` *input.fits output.fits*

6 データを見る

二次元画像を直接見るには、SAOIMAGE (ds9) を使います。
まず、IRAF を起動しているのとは別のターミナルで、SAOIMAGE を起動します：[3]

```
r22{aokiwk}11: ds9 &
```

そして、IRAF のターミナルから、タスク 'display' で画像を表示します。ここでは、'H4998.fits' というデータを表示してみます：

```
cl> display H4998.fits 1
z1=1420. z2=1967.862
```

表示の範囲等の調整は、display のパラメータとして与えることができます。

```
cl> epar display
```

と実行すると、次ページのような画面が出てきます：

[3] コマンドのうしろに '&' をつけておくと、いわゆるバックグラウンドジョブになり、このターミナルを引き続き別の作業に使うことができるので便利です

```
PACKAGE = tv
   TASK = display

image   =        ubc00478.fits  image to be displayed
frame   =                   1  frame to be written into
(bpmask =                 BPM) bad pixel mask
(bpdispl=                none) bad pixel display (none|overlay|interpolate)
(bpcolor=                 red) bad pixel colors
(overlay=                   ) overlay mask
(ocolors=               green) overlay colors
(erase  =                 yes) erase frame
(border_=                  no) erase unfilled area of window
(select_=                 yes) display frame being loaded
(repeat =                  no) repeat previous display parameters
(fill   =                 yes) scale image to fit display window
(zscale =                 yes) display range of greylevels near median
(contras=                0.25) contrast adjustment for zscale algorithm
(zrange =                 yes) display full image intensity range
(zmask  =                   ) sample mask
(nsample=                1000) maximum number of sample pixels to use
(xcenter=                 0.5) display window horizontal center
(ycenter=                 0.5) display window vertical center
(xsize  =                  1.) display window horizontal size
(ysize  =                  1.) display window vertical size
(xmag   =                  1.) display window horizontal magnification
(ymag   =                  1.) display window vertical magnification
(order  =                   0) spatial interpolator order (0=replicate, 1=linea
(z1     =                   ) minimum greylevel to be displayed
(z2     =                   ) maximum greylevel to be displayed
(ztrans =               linear) greylevel transformation (linear|log|none|user)
(lutfile=                   ) file containing user defined look up table
(mode   =                  ql)
```

ここで、xmag、ymag といったパラメータを変更すると、表示範囲などが変わります。この画面は、コロン (:) を入力し、さらに'q' を入力すると保存、終了します。パラメータ変更後、そのまま実行したいときは、':'、'go' とやるとタスクが実行されます。

SAOIMAGE の使い方についてはここでは省略します。

次に、データの断面図を見てみましょう。スペクトルデータの場合、通常このほうが有効です。これには 'implot' を使います。ここでは、フラットのデータを例にとります。

```
cl> implot H4998.fits
```

ここで、 ウインドウがあらわれて、まずはスリット方向の断面が示さ波長分散方向の断面図が示されます (図 4、ただしここでは一部のみ示している)。最初に表示される図の横軸を見ると、0–2050 になっていて、これはスリット方向のピクセル数を表しています。高い値の部分が光のあたっているところす。フラットデータの場合、スリット全面がランプによって照らされた場合のデータですので、この値の高い範囲がスリット長に相当します。縦、横のスケールの変更は、':' を入力後に、たとえば'x 500 1500' といった具合いに入力すると、500–1500 ピクセルについて表示されます。これも含めて、詳しくは、'?' でヘルプを見てください。(ヘルプは、' q' を 2 回入力すると抜けられます。)

なお、画面上で 'c' (=column) を入力すると、カーソルのおかれていた部分について垂直方向（この場合は波長分散方向）の断面が示されます。'l'(=line) を入力すると、再びスリット長方向の断面が示さ

れます。

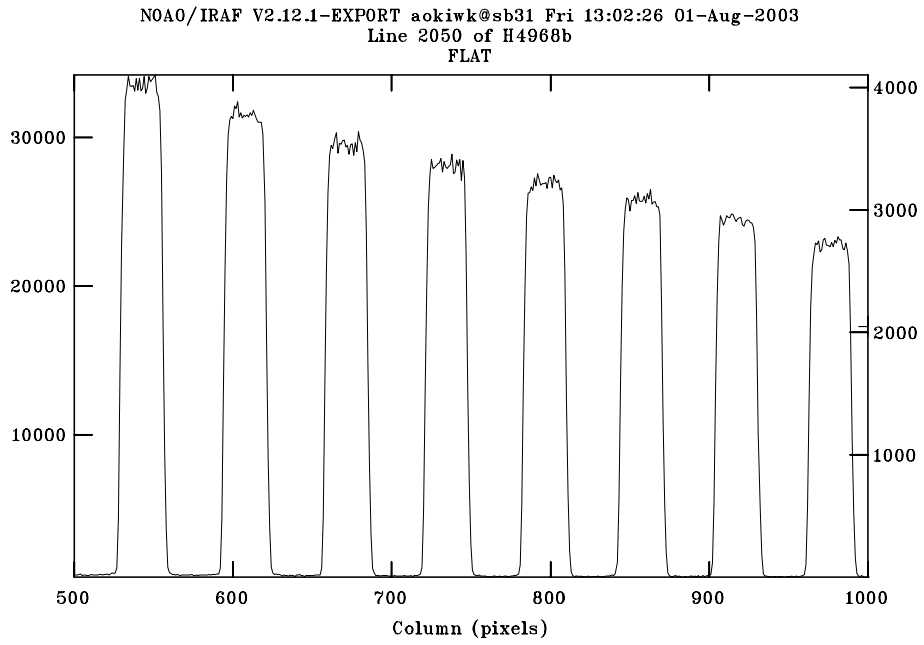


図 4: フラットデータの断面図 (スリット長方向) の一部

7　バイアス・ダーク補正

7.1　バイアス補正

CCD は、その読み出しの特性から、本来カウントゼロになるべき場合に値がゼロにならず、ゼロ点がずれています。一般にはその値はピクセルごとに少しずつ違います。そのため、露出を全くせず、読み出しだけ行って取得したフレーム（バイアスフレーム）をゼロ点と考え、このフレームを他のデータから差し引くことで、ゼロ点補正を行います。

まず、5.1 節で選んだバイアスデータ 3 枚から、バイアスフレームを作成します。精度をあげるために、このデータの median をとることにします（平均値でもよいのですが、CCD データでは、宇宙線ノイズと呼ばれる、ときおりとびぬけて高い値を示すノイズがあるので、median をとることにします)。'eparam imcombine' を実行すると、以下のように表示されます。

```
                                    I R A F
                      Image Reduction and Analysis Facility
PACKAGE = immatch
   TASK = imcombine

input   =             @bias_in  List of images to combine
output  =                 bias  List of output images
(headers=                     ) List of header files (optional)
(bpmasks=                     ) List of bad pixel masks (optional)
(rejmask=                     ) List of rejection masks (optional)
(nrejmas=                     ) List of number rejected masks (optional)
(expmask=                     ) List of exposure masks (optional)
(sigmas =                     ) List of sigma images (optional)
```

```
(logfile=               STDOUT) Log file

(combine=               median) Type of combine operation
(reject =                 none) Type of rejection
(project=                   no) Project highest dimension of input images?
(outtype=                 real) Output image pixel datatype
(outlimi=                     ) Output limits (x1 x2 y1 y2 ...)
(offsets=                 none) Input image offsets
(masktyp=                 none) Mask type
(maskval=                   0.) Mask value
(blank  =                   0.) Value if there are no pixels

(scale  =                 none) Image scaling
(zero   =                 none) Image zero point offset
(weight =                 none) Image weights
(statsec=                     ) Image section for computing statistics
(expname=                     ) Image header exposure time keyword

(lthresh=                INDEF) Lower threshold
(hthresh=                INDEF) Upper threshold
(nlow   =                    1) 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=                   0.) ccdclip: CCD readout noise (electrons)
(gain   =                   1.) ccdclip: CCD gain (electrons/DN)
(snoise =                   0.) ccdclip: Sensitivity noise (fraction)
(sigscal=                  0.1) Tolerance for sigma clipping scaling correction
(pclip  =                 -0.5) pclip: Percentile clipping parameter
(grow   =                   0.) Radius (pixels) for neighbor rejection
(mode   =                   ql)
```

ここで、input に 'H4946,H4948,H4950,H4952,H4954' を入れ（カンマ区切り、'.fits' は省略可）、output は 'bias' のようにします[4] 。combine のところは、median にします。これで ':go' で実行すると、バイアスフレーム ('bias.fits') ができます。おかしなことがおこっていないかどうか、データを、implot で確認しておきます。

できたバイアスフレームを他のすべてのデータから差し引き、バイアス補正を行います。2 次元データの演算には、'imarith' を用います。

```
cl> imarith H4998.fits - bias.fits H4998b.fits
```

ここでは、バイアスを差し引いた結果を、H4998b.fits というファイルに書き出しています。バイアスデータ以外のデータを、同様に処理しておきます。

7.2 ダーク補正

CCD は、シャッターを閉じて光が当たらない状態でおいても、時間がたつにしたがって、少しずつカウントがのります。これを暗電流（dark current = ダーク）とよびます。通常、その値は大きくはなく、

[4] H4946 などのファイル名を書いたファイル（1 行に 1 ファイル名）を用意しておくと、これらを一度に読み込むことができます。たとえば、bias_in という名前のファイルに上の 5 つのファイル名を書いておき、imcombine の input に`@bias_in` と指定すると、H4946 以下の 5 つのファイルが input として扱われます。

短い露出時間なら無視可能な場合も少なくありません。このダークの補正には、シャッターを閉じた状態で、天体データの露出と同じ時間の「露出」を行って取得した、ダークデータを、天体データから差し引くことで行います。

今回の実習で用いる観測データについては、露出時間が短く、ダークは高くないことがわかっているので、ダーク補正は省略します。

8 まずはスペクトルを抽出してみよう

このあと、フラットフィールディングと背景光の除去という作業がありますが、まず一度天体データから一次元のスペクトルデータを抽出してみましょう。[5] 。これによって、スペクトル抽出作業のための参照データをつくります。

二次元の CCD データから、横軸ピクセル数、縦軸カウント値のスペクトルを抽出するには、'apall' というタスクを用います。いろいろパラメータを設定する必要があるので、

```
ec> epar apall
```

とします。パラメータの設定例は、以下のとおりです。

```
                                I R A F
                  Image Reduction and Analysis Facility
PACKAGE = echelle
   TASK = apall

input   =               H4998b  List of input images
(output =            H4998b_ec) List of output spectra
(apertur=                     ) Apertures
(format =              echelle) 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 =                  yes) Resize apertures?
(edit   =                  yes) Edit apertures?
(trace  =                  yes) Trace apertures?
(fittrac=                  yes) Fit the traced points interactively?
(extract=                  yes) Extract spectra?
(extras =                   no) Extract sky, sigma, etc.?
(review =                  yes) Review extractions?

(line   =                INDEF) Dispersion line
(nsum   =                  100) Number of dispersion lines to sum or median

                                # DEFAULT APERTURE PARAMETERS

(lower  =                   -5.) Lower aperture limit relative to center
(upper  =                    5.) Upper aperture limit relative to center
```

[5] 「一次元」とよんでいるのは、元々の CCD データには、波長方向と空間（スリット）方向の二次元の情報が入っていたのに対し、空間方向のデータを足し合わせてしまうことで、波長方向一次元のデータとしてしまうためです。ここであつかう星のような点光源の場合には、空間方向の情報は意味がないために、足し合わせてカウントを稼ぐほうが有効です。一方、空間的に広がった天体では、空間方向の情報も重要ですから、単純に空間方向を足し合わせてしまってはいけない場合もあります。

```
(apidtab=                     ) Aperture ID table (optional)

                                # DEFAULT BACKGROUND PARAMETERS

(b_funct=             chebyshev) Background function
(b_order=                     1) Background function order
(b_sampl=          -10:-6,6:10) Background sample regions
(b_naver=                    -3) Background average or median
(b_niter=                     0) Background rejection iterations
(b_low_r=                    3.) Background lower rejection sigma
(b_high_=                    3.) Background upper rejection sigma
(b_grow =                    0.) Background rejection growing radius

                                # APERTURE CENTERING PARAMETERS

(width  =                    5.) Profile centering width
(radius =                   10.) Profile centering radius
(thresho=                    0.) Detection threshold for profile centering

                                # AUTOMATIC FINDING AND ORDERING PARAMETERS

nfind   =                    22  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 =                  -10.) Lower aperture limit relative to center
(ulimit =                   10.) Upper aperture limit relative to center
(ylevel =                   0.1) Fraction of peak or intensity for automatic wid
(peak   =                   yes) Is ylevel a fraction of the peak?
(bkg    =                    no) Subtract background in automatic width?
(r_grow =                    0.) Grow limits by this factor
(avglimi=                   yes) 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 bef
(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=                     0) 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=                 none) Extraction weights (none|variance)
(pfit   =               fit1d) Profile fitting type (fit1d|fit2d)
(clean  =                   no) Detect and replace bad pixels?
(saturat=               INDEF) Saturation level
(readnoi=                    8) Read out noise sigma (photons)
(gain   =                   1.) Photon gain (photons/data number)
(lsigma =                   4.) Lower rejection threshold
(usigma =                   4.) Upper rejection threshold
(nsubaps=                    1) Number of subapertures per aperture
(mode   =                   ql)
```

input には天体のデータを、output には好きな名前をいれます。

このタスクは、データの断面を切って（図 2 参照）、スペクトルがどの位置に写っているか探します (find)。そして、個々のスペクトルの断面上での位置を調べ (recent)、データを切り出す幅（アパーチャと呼びます）を決めます (resize)。その結果を画面に表示し、必要ならインタラクティブに修正します (edit)。これらをすべて行う場合には、’ find’,’recente’,’resize’,’edit’ をすべて’yes’ にしておきます。

アパーチャのサイズと位置が決まったら、個々のスペクトルのトレースを行い、その範囲でスリット長方向にデータを足し上げます。これらを行うためには、’ trace’,’fittrac’,’extract’ を’yes’ にしておきます。

設定すべき重要なパラメータとしては、

- # AUTOMATIC FINDING ... の ’nfind’: スペクトルを何本検出するか
- # RESIZING PARAMETERS の ’peak’ と ’ylevel’: ’peak’ を yes にすると、アパーチャサイズを決める際に、スペクトルのピークに対してどのくらいの高さのところでアパーチャを切るか、指定することができます。 ’peak’ を yes にした場合に、たとえば ’ylevel’ を 0.2 とすると、ピークの 20%のところでアパーチャサイズを決めることになります。

そのほか、詳しくは述べませんが、’extras’,’bkg’ は no に、’avglimi’ を yes にしておいたおうが無難です。

さて、このタスクを実行し、いろいろと尋ねられるのに答えていくと、画面に図 5 のような図が表示されます。これは自動的にアパーチャを決めた様子を示しています。ここでアパーチャの位置とサイズが適切か判断します。

問題があった場合は、アパーチャの位置やサイズを手で修正することは可能です[6] が、一度終了して、’apall’ のパラメータを変更してやり直した方がよいでしょう。

うまくいっているようなら、’q’ を入力すると、スペクトルのトレースが始まります。まずは図 6 のようにトレースの様子が表示されます。これは横軸が波長分散方向、縦軸がスリット長方向にトレースしたスペクトルの位置を’+’ マークで示し、それに関数フィットをした様子を示しています。縦軸、横軸とも、単位は CCD のピクセル数です。

このフィットが良くない場合には、関数の次数をあげたり、フィットをかける範囲を変更したりします。関数の次数の変更には、画面上で ’:order 3’ のように入力します。また、フィットをかける範囲は、

[6] 画面上、上のほうに示されるアパーチャのところにカーソルを持っていって’d’ を押すとアパーチャが削除されます。また、指定したい場所にカーソルをもっていって’n’ を押すと新たにアパーチャが指定されます。このような作業を行うと、一般にアパーチャの順番が狂ってしまうので、番号の付け直しを行います。そのためには、一番目として指定したいアパーチャ位置にカーソルをもっていき、’o’ を押します。すると’Aperture (1) =’ のように尋ねてくるので、’1’ を入力すると、そこを一番目にして順番に番号を割り振り直してくれます。

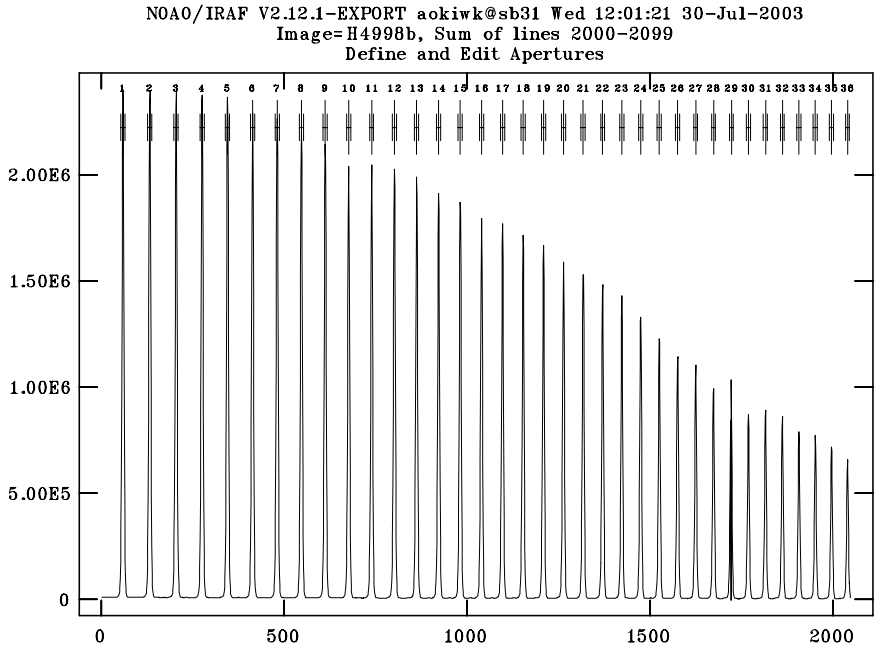


図 5: データのスリット長方向の断面図。apall で自動的にアパーチャ(上のほうに番号つきで表示されている) を決めたところ

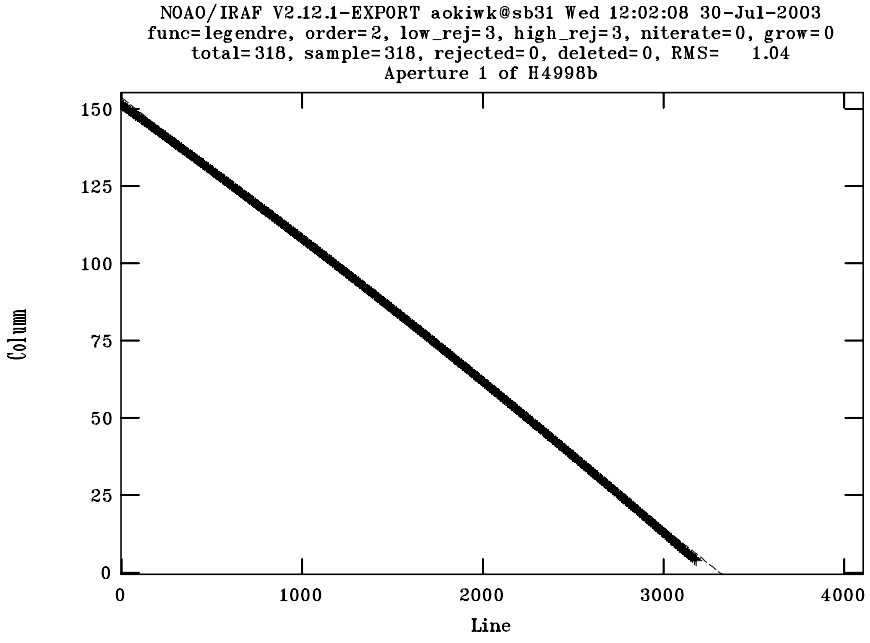


図 6: スペクトルのトレース作業。横軸が波長分散方向、縦軸がスリット長方向 (ピクセル数)。データ点が×印（重なっていて見にくいが）で、フィットした曲線が破線で示されている。この場合、破線が端のほうでデータ点からずれているので、もう少し次数を上げたほうがよい

画面上 2 箇所で 's' を入力することで指定します。フィットをしなおすには、'f' を入力します。(範囲指定を全体に戻すには't' を入力します。)

満足したら'q' を入力し、次の次数に進みます。

これらの作業が終わると、とりあえずスペクトルができます（図 7)。

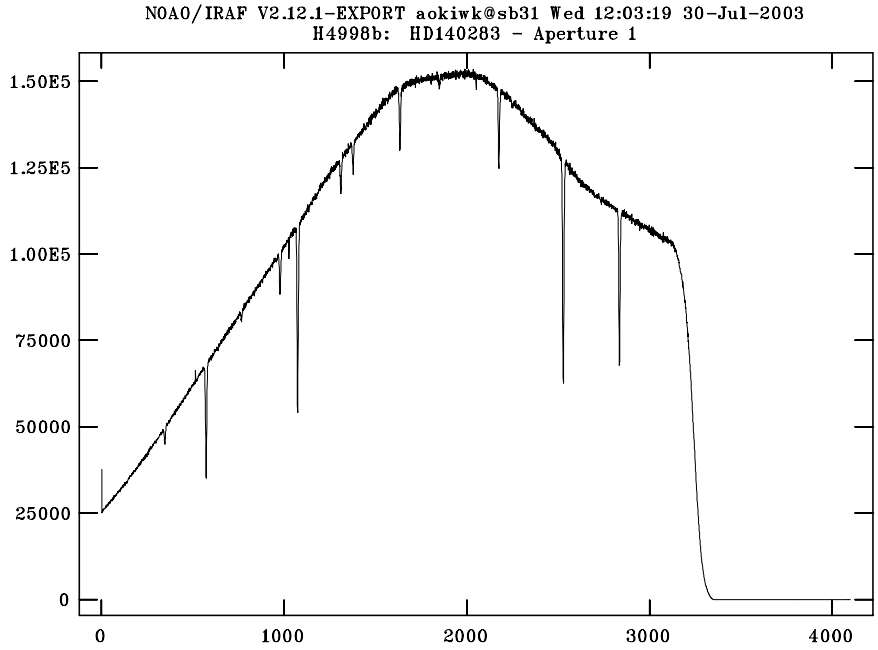


図 7: apall で得られたスペクトル。この段階では横軸はピクセル数のままである

スペクトルを簡単に表示するには、'splot' というタスクが便利です。詳しくは 16 章で説明します。

9 フラットフィールディング

CCD は、飽和レベルにいたるまでは、照射された光量に比例したカウント（もともとは電子数）を記録します。その線形性は優れており、それが写真乾板にかわって CCD が用いられるようになった理由のひとつです（もちろん効率がよいのが最大の理由です)。しかし、CCD の感度（上の比例係数）は、ピクセルごとに少しずつ違います。これを補正するのがフラットフィールディングとよばれる作業です。

この補正のためには、天体データを取得したのと同じ波長設定で、連続光を出すランプの光をスリット全面にあて、データを取得しておきます（フラットデータ。図 4)。これにより、波長依存性も含めた感度特性を記録した較正用データができます。この特性は時間変化（CCD の温度の変化などによると思われる）がありうる場合には、天体データを取得したのとできるだけ近い時間に取得することが望ましいといえます。

バイアス補正 (およびダーク補正) を行ったフラットデータから、バイアスデータの時と同じように median をとって、フラットフレーム ('flat.fits') をつくります。用いるタスクは、同じく 'imcombine' です。やりかたは省略します。

フラットフィールディングは、感度の比例係数の補正を行うわけですから、天体データを、フラットフレームで割ることになります。割り算も、'imarith' を用いて行うことができます。しかし、その前に、フラットフレームの規格化をしておきましょう。

通常、フラットデータは、飽和レベルに近いところまで光を当てて、高い S/N 比で取得してあります。これで天体データを割ってしまうと、非常に小さい (1 以下の) 値になってしまいます。以後の解析では、カウントの絶対値はあまり重要ではありませんが、天体のカウント数の情報が失われてしまうというのは好ましくありません。また、エシェルの干渉次数の異なるスペクトルの間には、スリット長からはずれて光が当たっていない部分があります。そこのカウントはゼロに近く、フラットデータでそのまま割ると、この部分は非常にノイズの高いデータになってしまいます (後の背景光除去ができなくなってしまいます)。
[7]

そこで、フラットフレームのカウントが、全体の平均が 1 になるように規格化しておきます。規格化には、'apnormalize' を用います。パラメータの設定は以下のようになります。

```
                                  I R A F
                     Image Reduction and Analysis Facility
PACKAGE = echelle
   TASK = apnormalize

input   =                  flat  List of images to normalize
output  =                 flatn  List of output normalized images
(apertur=                      ) Apertures
(referen=                H4998b) List of reference images

(interac=                   yes) Run task interactively?
(find   =                    no) Find apertures?
(recente=                   yes) Recenter apertures?
(resize =                   yes) Resize apertures?
(edit   =                   yes) Edit apertures?
(trace  =                    no) Trace apertures?
(fittrac=                    no) Fit traced points interactively?
(normali=                   yes) Normalize spectra?
(fitspec=                   yes) Fit normalization spectra interactively?

(line   =                 INDEF) Dispersion line
(nsum   =                    10) Number of dispersion lines to sum or median
(cennorm=                    no) Normalize to the aperture center?
(thresho=                   10.) Threshold for normalization spectra

(backgro=                  none) Background to subtract
(weights=                  none) Extraction weights (none|variance)
(pfit   =                 fit1d) Profile fitting type (fit1d|fit2d)
(clean  =                    no) Detect and replace bad pixels?
(skybox =                     1) Box car smoothing length for sky
(saturat=                 INDEF) Saturation level
(readnoi=                    0.) Read out noise sigma (photons)
(gain   =                    1.) Photon gain (photons/data number)
(lsigma =                    4.) Lower rejection threshold
(usigma =                    4.) Upper rejection threshold
```

[7] フラットのデータは、人工光源（ハロゲンランプ）を分光器の前において、天体と同じように「観測」することで得られます。光源は、大局的にみると強度に波長依存性があり、たいていは短波長側で弱くなります。観測波長域が広い場合、CCD の飽和レベル以下でデータを取ろうとすると波長によってはかなり光量が低くなってしまいます。その場合、光源側にフィルターを用いてエネルギー分布 (波長依存性) を調整するか、各波長域で最適となる設定（露出時間や光源の電流）で複数回にわけてフラットのデータを取得する必要があります。HDS の場合、たいていは 2 つの CCD それぞれで最適になるように、2 種類の設定でフラットのデータをとっています。整約の際には、適切なほうを選択して使います。

```
(functio=              spline3) Fitting function for normalization spectra
(order  =                    3) Fitting function order
(sample =                    *) Sample regions
(naverag=                    1) Average or median
(niterat=                    5) Number of rejection iterations
(low_rej=                  3.0) Lower rejection sigma
(high_re=                  3.0) High upper rejection sigma
(grow   =                  0.0) Rejection growing radius
(mode   =                   ql)
```

'apall' の場合と似ていますが、これは規格化を行う際に、一度スペクトルの抽出を行って、それから波長方向の光の強度分布に対して関数をフィットするためです。

ただし、ここでは、さきほどスペクトルの抽出を行った天体データ (H4998) を参照データとして使うことにします。天体とフラットは、同じ分光器の設定でとっているため、スペクトルの本数や位置は基本的に同じです。したがって、先ほどの天体データを参照することで、作業を簡略化することができます[8] 。

天体データを参照するには、referen にファイル名を書きます。スペクトルの位置探し (find) とトレース (trace および fittrac) で参照データを利用することにして、これらを no にしておきます。アパーチャの位置とサイズは、天体とフラットで異なりますので、これらはフラットデータ自身から決めるべきですから、yes にしておきます。アパーチャのサイズを決めるための 'peak' と 'ylevel' というパラメータが apnormalize にはありません。これらは、別途 'apresize' というタスクの中で指定する必要があります (apnormalize は、実行の際に apresize を呼び出しています)。apresize のパラメータの例を参考までに示しておきます。

```
                                  I R A F
                       Image Reduction and Analysis Facility
PACKAGE = echelle
   TASK = apresize

input   =                         List of input images
(apertur=                        ) Apertures
(referen=                        ) Reference images

(interac=                       no) Run task interactively?
(find   =                      yes) Find apertures?
(recente=                       no) Recenter apertures?
(resize =                      yes) Resize apertures?
(edit   =                      yes) Edit apertures?

(line   =                   INDEF) Dispersion line
(nsum   =                       1) Number of dispersion lines to sum or median
(llimit =                   -20.0) Lower aperture limit relative to center
(ulimit =                    20.0) Upper aperture limit relative to center
(ylevel =                     0.4) Fraction of peak or intensity for automatic wi
(peak   =                      yes) Is ylevel a fraction of the peak?
(bkg    =                       no) Subtract background in automatic width?
(r_grow =                      0.0) Grow limits by this factor
(avglimi=                      yes) Average limits over all apertures?
```

[8] 天体のスペクトルの写っている位置情報などは、database というディレクトリの下に、apH4998 というようなファイルとして保存されています。なので、database 以下のファイルを消してしまったり、参照データを別のディレクトリにコピーして database 以下との対応がつかなくなってしまうとデータを参照できなくなるので注意してください。

```
(mode   =                    ql)
```

さて、apnormalize を実行すると、途中までは apall の場合と同じように進みます。トレースの部分を、参照データを使ってしまっているので、この部分は手作業がありません。トレースが終わると、図 8 のように、フラットのスペクトルが表示されます。

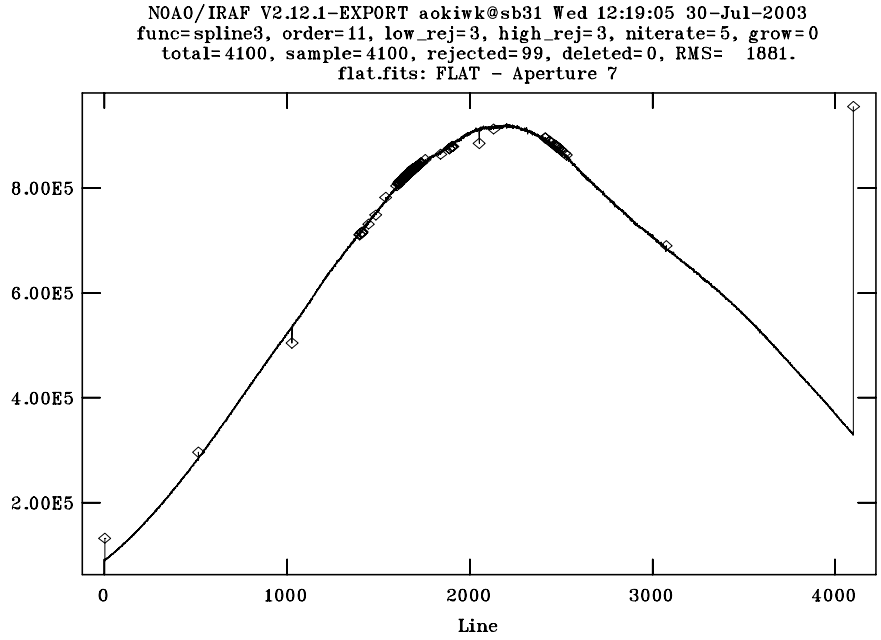


図 8: フラットのスペクトルへの関数フィット

ここで、データに対しては、関数フィットが行われているはずです。関数の次数を変えるなどして、フィットを良くします。関数の種類を変えるには、画面上で ':func spline3' のように、その次数を変えるには ': order 9' のように入力します。また、必要に応じて、関数フィットを行う範囲を指定します。画面上で 2 箇所's' を入力すると、その間がフィッティングを行う範囲として指定されます。複数箇所指定可能です。その後で'f' を入力すると実際にフィットが行われます。(指定を取り消して全体に対するフィットに戻すには、画面上で 't' を入力します。)

終ったら 'q' を入力すると次の次数のスペクトルに移ります。この作業をすべての次数に対して行います。

終了したら、できたファイルを、implot でみてみましょう。問題なくできれいれば、図 9 のようになります。値がピッタリ 1 になっている部分は、アパーチャの外、つまりスペクトルとスペクトルの間の光の当たっていない部分です。アパーチャ内の値のばらつきが、CCD の感度ムラを表しています。

こうして規格化したフラットフレームで、天体データを割ります。割り算にも、以下のように `imarith` が使えます。

```
imarith H4998b / flatn H4998bf
```

これでフラットフィールディングは完了です。

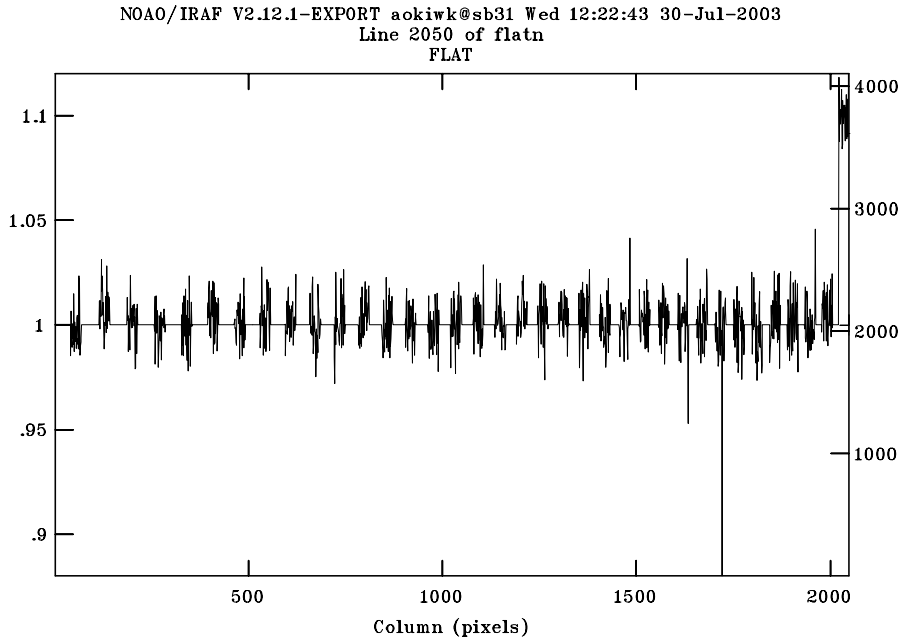


図 9: 規格化されたフラットの断面図 (スリット長方向)

10 背景光の除去

天体データの断面図を見ると、散乱光などがあるため、スペクトル間のカウントは一般にゼロになっていません (図 10)。これを背景光として除去する必要があります。これには、天体データについてアパーチャを決め、光の当たっている部分をマスクしてしまい、それ以外の部分を背景光と考えて曲面フィットして天体データから差し引きます。この作業は、天体データに対してのみ行います。

タスクは apscatter です。パラメータの例を以下に示しました。apnormalize のときと同様に、'apfind', 'aprecenter', 'apresize', 'apedit', 'aptrace' のパラメータ指定を行います。'eparam apscatter' とやって、'apscat1' (または 'apscat2') のところで ':e' を入力すると、dispersion (または slit) 方向のフィッティングに用いる関数を指定できます。タスクを実行すると、apnormalize のときと同様に、天体データのスペクトルのトレースを行い、その後に、バックグラウンドに対して局面フィットした結果が、slit 方向について示されます（図 11)。満足できたら 'q' で抜けます。別の line についてのフィットも見たければ、'line 200' などと入力します。slit 方向について満足したら、' quit' とやると、今度は dispersion 方向のフィットが示されます（図 12)。こちらは、'column 2500' などでいろいろな column についてフィットの様子を見ることができます。満足したら'quit'。しばらくすると、バックグラウンドを差し引いた画像が得られます。

```
                                  I R A F
                    Image Reduction and Analysis Facility
PACKAGE = echelle
   TASK = apscatter

input   =              H4998bf  List of input images to subtract scattered lig
output  =             H4998bfs  List of output corrected images
(apertur=                      ) Apertures
(scatter=                      ) List of scattered light images (optional)
(referen=               H4998b) List of aperture reference images
```

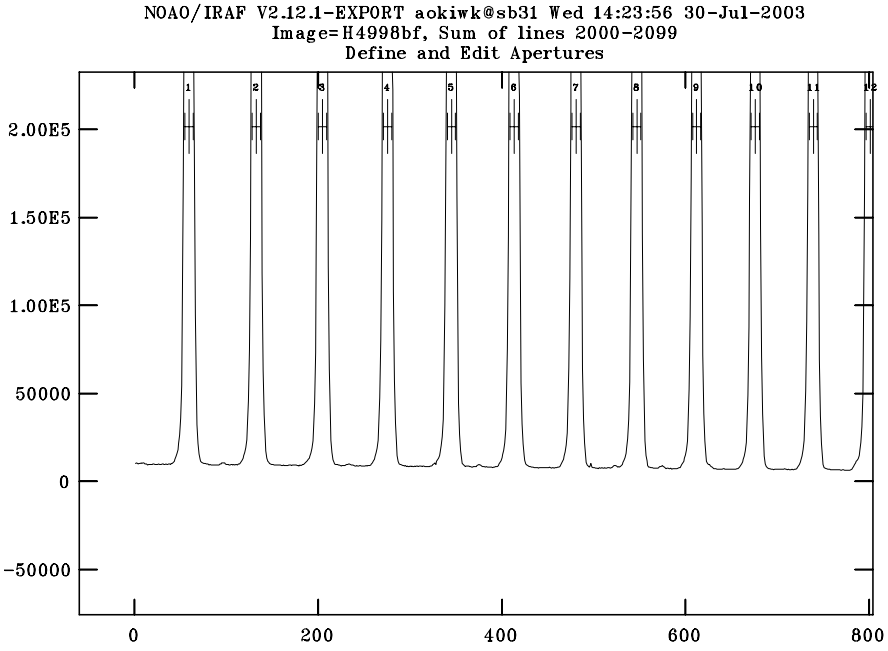


図 10: 天体データの断面図（apscatter を実行中に得られる図）

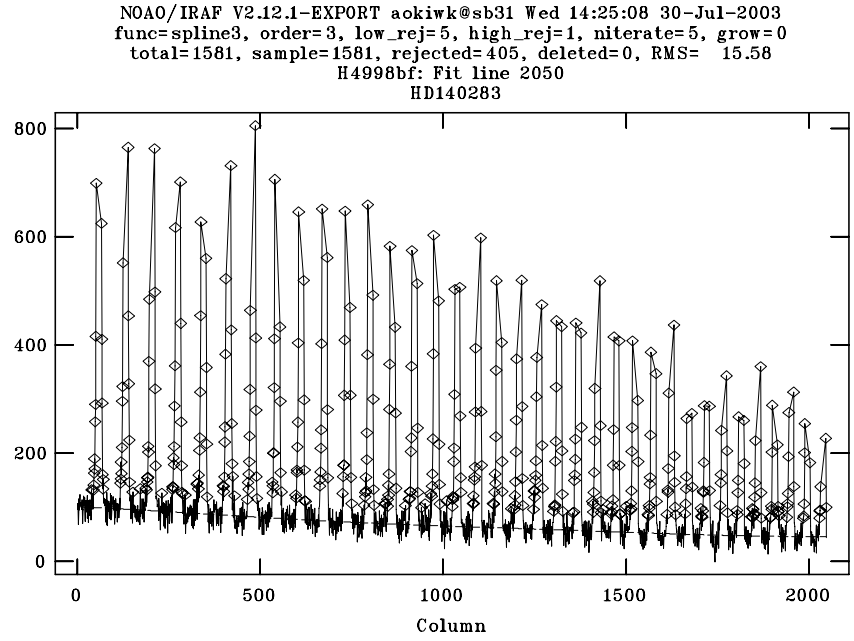


図 11: アパーチャ内をマスクしたデータの断面図（スリット長方向）。アパーチャ間のデータに関数フィットをかけている。

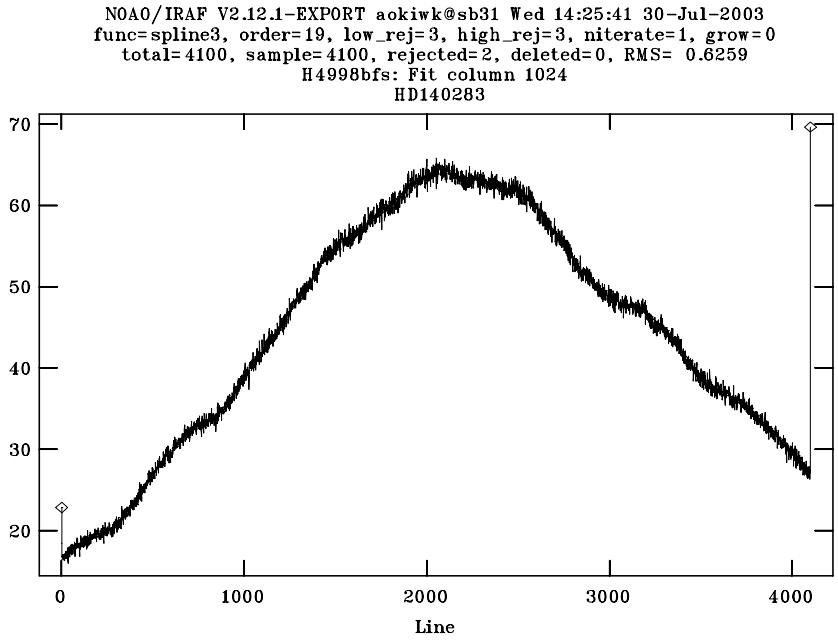


図 12: アパーチャ内をマスクしたデータの断面図（波長分散方向)。データに関数フィットをかけている。

```
(interac=                  yes) Run task interactively?
(find   =                   no) Find apertures?
(recente=                  yes) Recenter apertures?
(resize =                  yes) Resize apertures?
(edit   =                  yes) Edit apertures?
(trace  =                   no) Trace apertures?
(fittrac=                   no) Fit the traced points interactively?
(subtrac=                  yes) Subtract scattered light?
(smooth =                  yes) Smooth scattered light along the dispersion?
(fitscat=                  yes) Fit scattered light interactively?
(fitsmoo=                  yes) Smooth the scattered light interactively?

(line   =                INDEF) Dispersion line
(nsum   =                  100) Number of dispersion lines to sum or median
(buffer =                   1.) Buffer distance from apertures
(apscat1=                     ) Fitting parameters across the dispersion
(apscat2=                     ) Fitting parameters along the dispersion
(mode   =                   ql)
```

11 一次元スペクトルの抽出

こうして較正の済んだ天体データから、一次元スペクトルの抽出を行います

タスクは 8 節で用いた apall です。ただし今回は、apscatter の場合と同じく、最初にトレースを行ったデータを参照データとして用いてよいので、非常に簡単に行えます。apall のパラメータ指定は、以下のようにするとよいでしょう。ここでは、アパーチャの大体の位置ととトレースのしかたは参照データをそのまま使い、アパーチャの細かい位置とサイズ (これは観測ごとに、シーイングなどによって変動します) はデータ自体から決めています。

```
                                    I R A F
                        Image Reduction and Analysis Facility
PACKAGE = echelle
   TASK = apall

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

以上の処理では、スリット長方向のデータは、星像程度の範囲内で足しあわされることになります。しかし、分光器によっては、スリット像が CCD のピクセル並びに揃っていない場合があります。また、スリット像は一般に多かれ少なかれ曲がっているので、スリットが長くなると CCD のピクセルには揃いません。このような場合には、単純にスリット長方向のデータを足しあげるのではなく、スリット長方向に星像を分割し、それぞれの場所でスペクトルを切り出して処理する必要があります。

このためには、星像に対してアパーチャサイズを決める際に、複数のアパーチャ（サブアパーチャ）をとるように指定します。パラメータは nsubaps で、リストの最後のほうで以下のように指定します。

```
...
(nsubaps=                     5) Number of subapertures per aperture
...
```

この場合、5 つのサブアパーチャに分割していて、得られるスペクトルは図 13 のようになります。カウントが高いものが星像中央付近、低いものが星像の端のほうをトレースした結果と理解できます。

得られるデータには、サブアパーチャごとに番号 (この場合 1-5) がつけられ、別個にファイルができます。それぞれについて、このあと説明する波長較正を行い、その上で必要に応じてデータを足しあわせます。

12 波長較正

以上で得られたスペクトルは、まだ横軸が CCD のピクセル数になっています。これを波長スケールになおします。

この作業は、天体のデータ取得と同じ設定でとった比較光源（Th-Ar ランプ）データ（comparison）を用いて行います。Th 輝線の波長はあらかじめわかっていますので、このデータから、CCD のピクセル数と波長との関係式を導くことができます。

12.1 Th 輝線の同定

まず、comparison のデータを、天体のスペクトルと同じアパーチャで一次元スペクトルにします。それには、'apall' を実行する際に、'input' には comparison のファイル名を与え、referen(ce) に、先ほどの天体データ（まだ二次元画像の状態のもの）のファイル名を与えます。そして、' find','recente','resize'

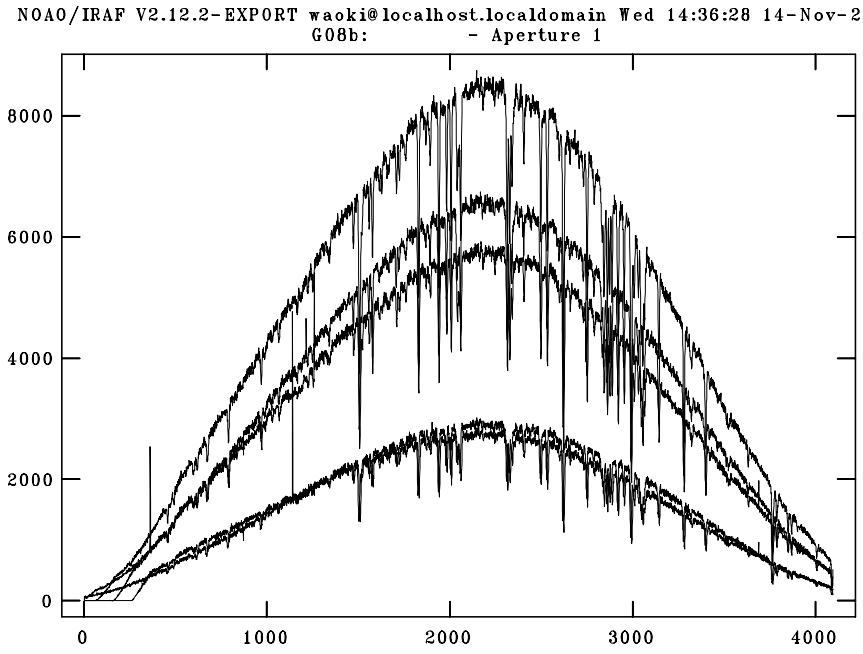


図 13: サブアパーチャに分割して apall を行った結果。5 つのサブアパーチャごとに抽出されたスペクトルが表示されている。

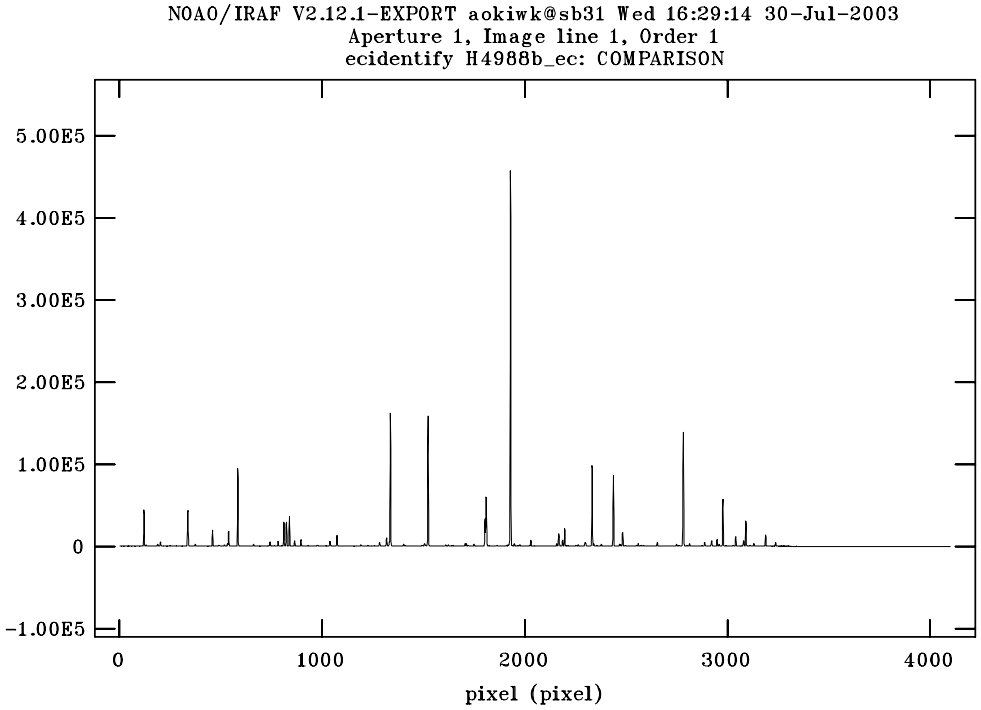


図 14: comparison スペクトルの例（ecidentify の途中で表示される図）

を no にして、実行します。質問に yes と答えていくと、referen で与えた天体のアパーチャを使って抽出した、comparison のスペクトルが得られます。

こうして得られた comparison スペクトルは、多数の Th (と Ar) の輝線を含んでいます。comparison のスペクトルの例を図 14 に示します。次は、この輝線の同定を行い、波長を与える作業を行います。'

その前に、データがどの程度の波長域をカバーしているか、知っておくと便利です。データのヘッダ部を見ると、以下のような 3 行があります。

```
WAVELEN =                480.60 / Center wavelength of the center order (nm)
WAV-MAX =                540.17 / Maximum wavelength recorded (nm)
WAV-MIN =                414.20 / Minimum wavelength recorded (nm)
```

つまり、このデータは、414.2–540.17 nm をカバーしているということです。ただし、これらの値は、あくまで目安です。

また、比較するための波長のつけられた Th のデータが必要です。HDS の場合、Th 輝線の波長同定を行ったアトラス[9] が準備されていますので、それを利用することにします。

12.2 Th 輝線の波長入力/波長スケールの作成

次に、同定した輝線の波長を計算機に読み込ませる作業を行います。これには、' 'ecidentify' を用います。'epar ecident' を実行して、パラメータ設定を行います。'images' には、もちろん comparison スペクトルを指定します。' そして、coodli' には、'linelists$thar.dat' を指定します。これは、IRAF パッケージの中に準備されている Th-Ar スペクトル線リストを参照することを意味します。

このタスクを実行すると、図 14 のようなスペクトルが表示されます。必要に応じて図は拡大してください（方法は'splot' の場合と同じ)。ここで、同定された輝線にカーソルを持っていき、'm' を入力します。望みの輝線の上に印（縦線）がついたら、その輝線の波長を入力します。0.1Å のレベルまで書き込んだら、リターンをしてかまいません。スペクトル線リストから波長を読んできてくれます（正しい波長を読んできてくれたかどうかは確認すること)。この作業を、同定した輝線について、繰り返し行います。' m' で輝線を指定するのに失敗したり、波長の入力を誤った場合には、その輝線にカーソルを持っていって、'd' を入力すると、消去してくれます。(画面上からただちには消えてくれない場合もあるので、そのときには 'r' を入力して画面の書き換えをしてやる必要があります。)

ひとつの次数に対して数本同定できたら、画面上で 'k' を入力して次の次数に進みます（戻るには 'j')。この作業を繰り返します。すべての次数について同定しなくても大丈夫ですが、できるだけデータ全体にムラなく作業を行っておくのが望ましいです。

こうして全体にわたって同定した輝線の波長を読み込ませたら、'f' を入力します。これで、フィッティングにより、ピクセル数と波長の関係式をつくります。図には、フィットした結果の残差が示されます。大きくはずれた点があったら、同定ミスかもしれませんので、確認してみて、ミスだったら消去します。消去の方法は、その点の近くにカーソルをもっていって 'd' を入れます。関数の次数は、さきほど、パラメータで与えましたが、この作業の途中でも、': xorder 4' ':yorder 3' というような具合に入力することで変えられます。図 15 は、波長同定が行われ、フィットもうまくできた場合の残差を示しています (最終結果なので、データ点は非常に多くなっています)。横軸は波長分散方向（ピクセル数）です。

また、横軸をスペクトルの次数にすることもできます。画面上で'x o' と入力します。

フィッティングがうまくいっているようなら（残差は 0.01 以下になっているはずです)、'q' を入力します。これで図 14 の状態に戻ります。ここで、'l'(エル) を入力すると、IRAF パッケージ内の Th-Ar スペ

[9] Honda & Aoki 2001, http://www.naoj.org/Observing/Instruments/HDS/

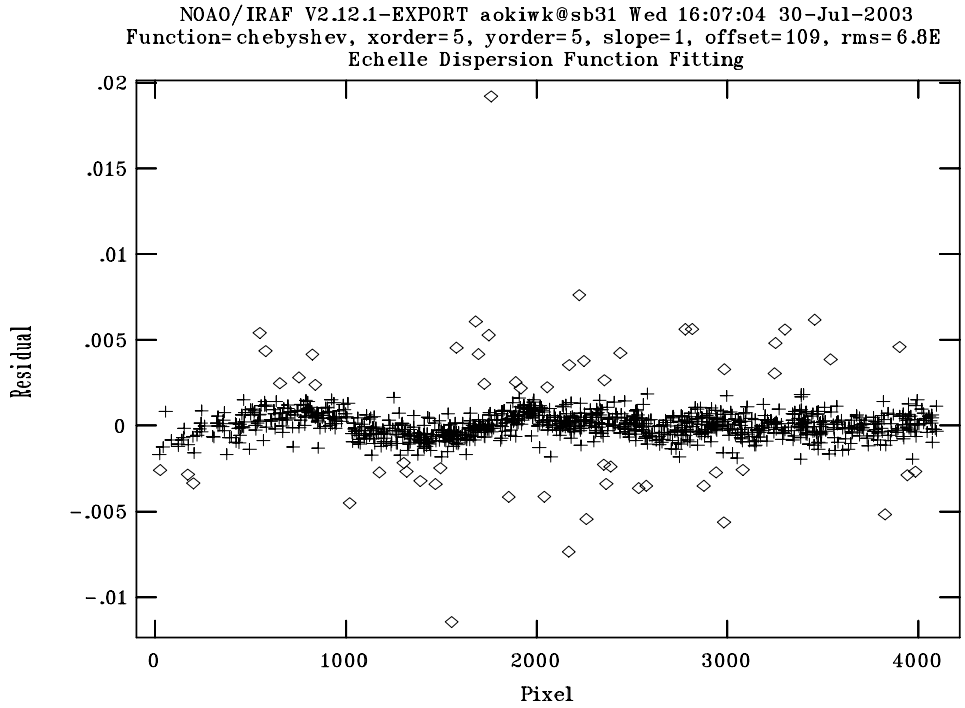


図 15: Th 輝線の同定・関数フィットを行った結果の残差。横軸は波長分散方向（ピクセル数、縦軸が残差（Å）

クトル線リストを参照して、他の輝線の同定も自動的に行ってくれます。うまくいっていれば、多数の輝線が同定されます。そこで再度 'f' を実行すると、今度は多数の線についてフィッティングを行った結果が表示されます。同様に、大きくはずれた点を消去するなどし、'q' で終了します。必要なら'l' での同定を繰り返します。うまくいっているようなら、図 14 の画面で'q' を入力し、ecidentify を終了します。

作成された波長スケールは、ディレクトリ database 以下の ec　で始まるファイルに書き込まれます。例えば、comp_ec1 というデータについては eccomp_ec1 というファイルが作られます。

さて、同じ設定で取得した comparison データでも、温度変化などによってデータに多少のずれが生じます。このような場合、上のようにして作成した波長スケールを参照して、別のデータについては自動的に波長スケールを作成する機能があります。そのタスクは ecreidentify です。ecidentify で波長スケールを作成したデータを comp_ec1, 同じ設定で取得した comparison を comp_ec2 とすると、

```
ec> ecreident comp_ec2 refe=comp_ec1
```

とすると、comp_ec2 についても波長スケールが作成されます。その際に、どの程度波長方向にシフトしたか、などの結果も表示されます。

12.3　天体データの波長較正

上でつくった波長スケールを、天体データに適用します。まず、タスク' refspectra' で、天体データに適用する波長スケールを指定します。

```
ec> refs H4998bfs_ec refe=H5008b_ec
```

ここで、H4998bfs_ec.fits が天体スペクトル、H5008b_ec.fits が comparison スペクトルです。質問には yes と答えます。

このタスクのパラメータには以下のようなものがあり、必要に応じて書きかえてください。まず、select によって、どの comparison データの波長スケールを用いるかを指定します。サブアパーチャに分割してスペクトルを抽出した場合には、match を指定します。

また、sort　や group は、そこに書き込まれた FITS ヘッダ情報を探しにいくので、データのヘッダに該当するものがないとエラーになります。HDS の場合、sort には ut や mjd を、group は空白にしておけばとりあえず動きます。

```
                                  I R A F
                     Image Reduction and Analysis Facility
PACKAGE = echelle
   TASK = refspectra

input   =                         List of input spectra
(referen=                 *.imh) 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   =                    mjd) Sort key
(group  =                      ) 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 spectr
(logfile=         STDOUT,logfile) List of logfiles
(verbose=                     no) Verbose log output?
answer  =                    yes  Accept assignment?
(mode   =                     ql)
```

次に、タスク 'dispcor' (dispersion correction) で、実際に天体スペクトルの波長較正を行います。

```
ec> dispcor H4998bfs_ec H4998bfs_ecw.fits
```

波長較正されたスペクトルが、H4998bfs_ecw.fits に保存されます。結果は図 16 のようになります。

13　コンティニュームの決定と規格化

以上で、横軸が波長になったスペクトルが得られました。次は、縦軸方向の処理です。[10]

[10] その前に、必要ならばスペクトルのトリミングを行っておきます。それぞれのスペクトルは、中央から離れるにしたがってカウントが低くなっています (図 16)。ここで扱っているデータの場合、特に短波長側のカウントが低くなっています。一方、隣あった次数のスペクトルを比べてみると、カバーしている波長範囲に重複があります。最終的には、これらを組み合わせて一本のスペクトルをつくることになりますが、その際に、あまりにカウントの低いデータを残しておくと、データの質を

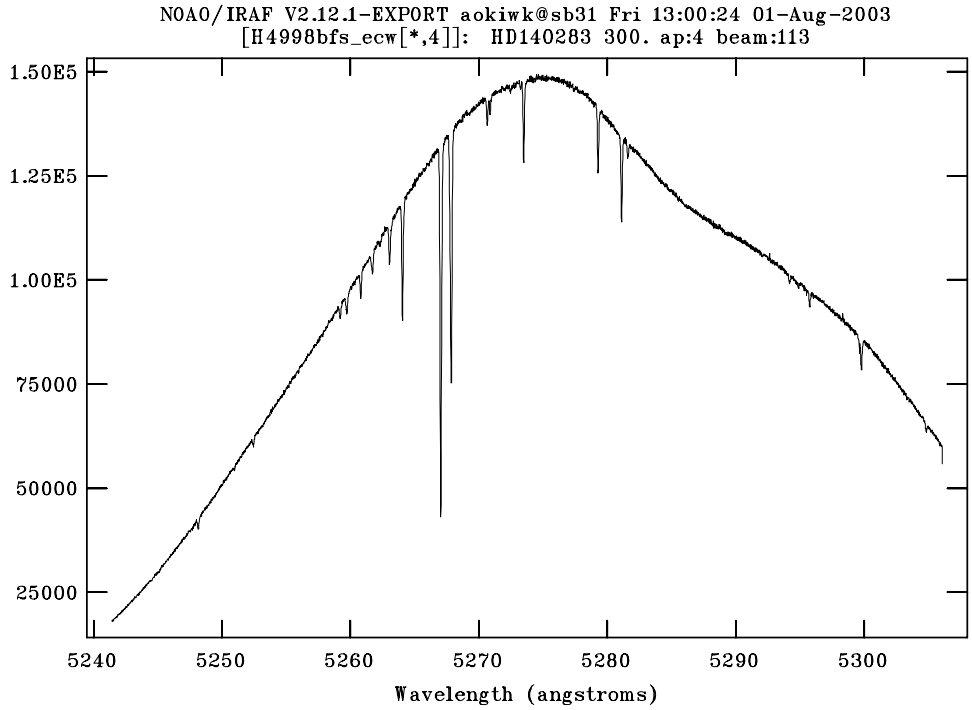


図 16: 波長較正を行った結果得られるスペクトル

スペクトルデータでも、フラックス較正をきちんと行って、縦軸をフラックス密度になおす場合があります。しかし、ここでは、そういった処理は行わず、連続光部分 (コンティニューム) で規格化することにします。これは、後の解析で、吸収線の波長を測定したり、吸収線の連続光成分に対する比を議論したりするためです。

コンティニュームの決定と規格化は、'continuum' を用いて行います。

```
ec> continuum H4998bfs_ecw H4998bfs_ecwc.fits
```

と実行すると、図 17 のように、吸収線と思われる部分をはじいて、コンティニュームを見積もってくれます。フィットする関数の変更は': func spline3' や ':order 5' のように入力して行います。フィットする範囲を指定したければ、その一方にカーソルを持っていって 's' を入力し、もう一方でもう一度's' を入力すると、下にその範囲が示されます。この方法で、複数ヶ所指定できます。指定を取りやめてもどしたいときには、't' を入力します。フィッティングの際にはじきたい点は、カーソルをもっていって、'd' と入力します。再度フィットを行うには、'f' を入力します。

満足したら、'q' を入力すると、次の次数に進みます。この作業を繰り返して、すべての次数に対してコンティニュームを決定します。タスクが終了すると、コンティニュームを 1 にしたスペクトルができあがります（図 18)。

落としてしまうことになります。そこで、スペクトルの短波長側を一部切り落とすことにします。以下の例は、短波長側を 600 ピクセル分切り落とす場合です。

imcopy H4998bfs_ecw[601:4100,*] H4998bfs_ecwt

また、複数回の露出を行ったり、サブアパーチャごとにスペクトルを抽出した場合で、データを足しあげたい場合は、この時点で行っておくとデータ処理の効率はよくなります。足しあげるためには scombine を用います。

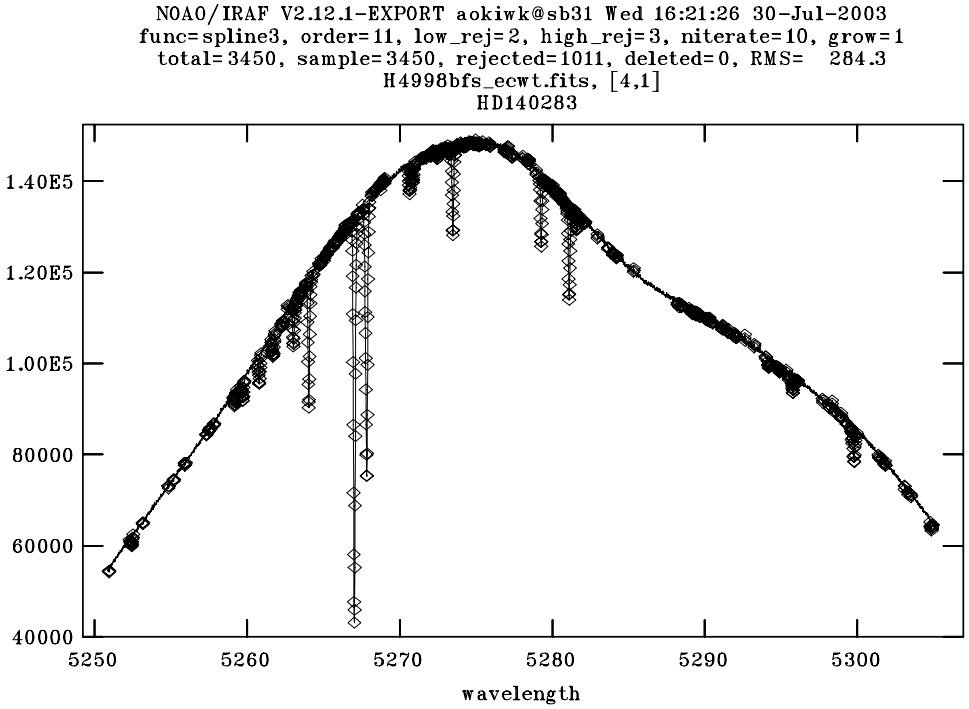


図 17: スペクトルへのコンティニュームフィッティングの例

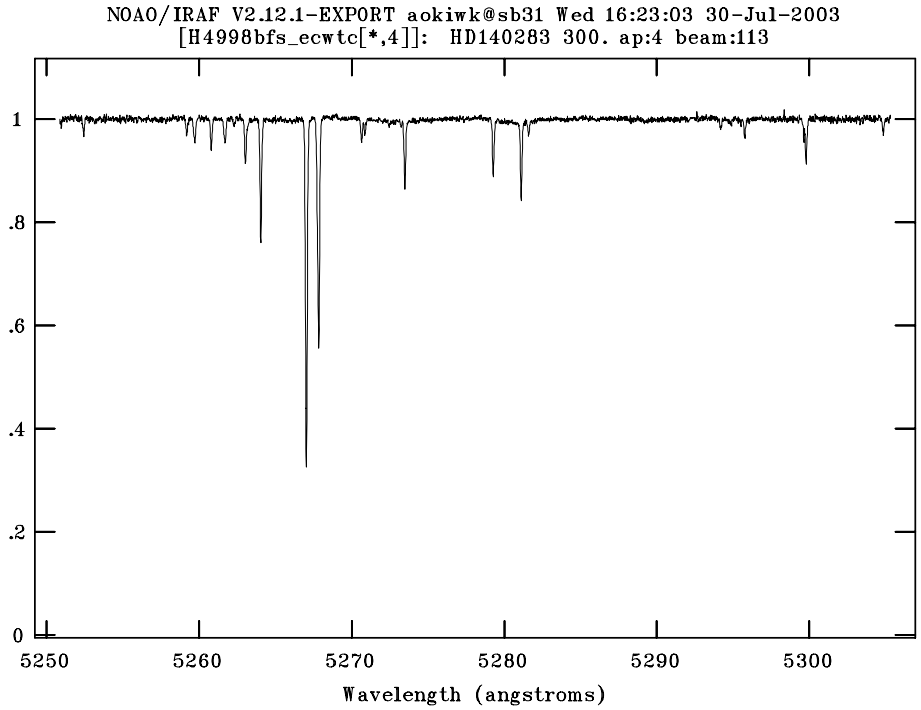


図 18: 規格化されたスペクトルの例

14　スペクトルの一本化

各オーダごとに規格化されたスペクトルは、scombine を用いてつなげることができます。波長が 2 つの次数でダブっている場合は、その平均をとるなどの処理が可能です。scombine では、input、output のファイル名を指定するほか、group は'images' とし、combine については 'average' などとします。[11]

しかし、二つのとなりあったオーダの規格化されたスペクトルを単純に平均してしまうと、カウント値の低いほうのノイズが大きく効いてきてしまうことがあります。たとえば、図 19 のようなスペクトルには、カウントがほぼゼロに落ちている部分があるので、これををそのまま処理すると、図 20 のようにデータの質の悪い部分ができてしまいます。

これを避けるためには、カウント値を考慮して複数のオーダのスペクトルを一本化する必要があります。まず、一次元化し、波長較正したスペクトル (図 19) にコンティニュームフィットを行い、規格化を行います (前節参照)。もとのデータを規格化したデータで割ると、コンティニュームのスペクトルが得られます (図 21)。これは continuum でフィットした関数そのものです。

それぞれのデータを scombine によって一本化します。この際に、パラメータは以下のようにして、平均ではなくて和をとります。

```
(group  =               images) Grouping option
(combine=                  sum) Type of combine operation
```

すると、天体データのほうからは、図 22 のような妙なスペクトルが得られます。しかし、コンティニュームレベルのデータを一本化したもの (図 23) でこれを割ると、きれいに規格化・一本化されたスペクトルが得られます (図 24)。

[11] なお、scombine は、複数のフレームから得られたスペクトルの足し合わせ（平均）にも用いられます。

input=A.ec,B.ec,C.ec

のように与え、group には 'aperture', combine には 'average' などと与えると、3 つのスペクトルの平均が得られる。各データの S/N 比に応じてウエイトをかけることもできます。

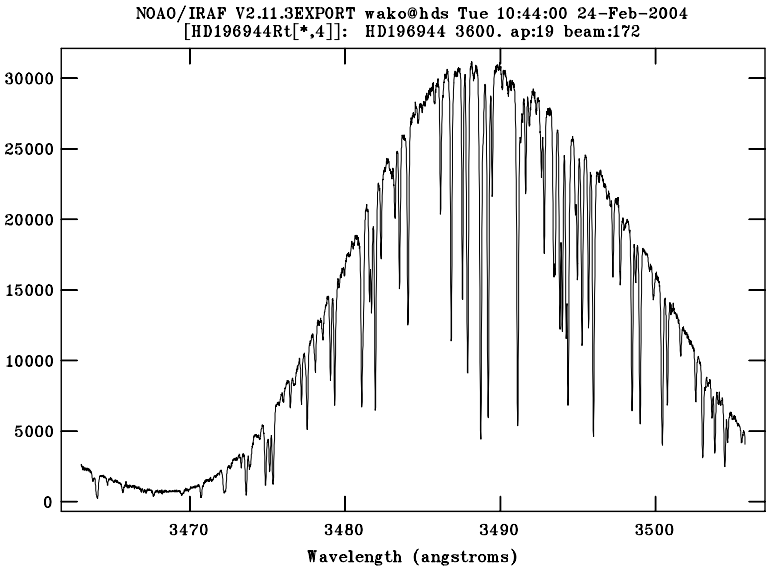


図 19: 波長較正まで行って得られたスペクトル。オーダの端ではカウントが非常に低い。

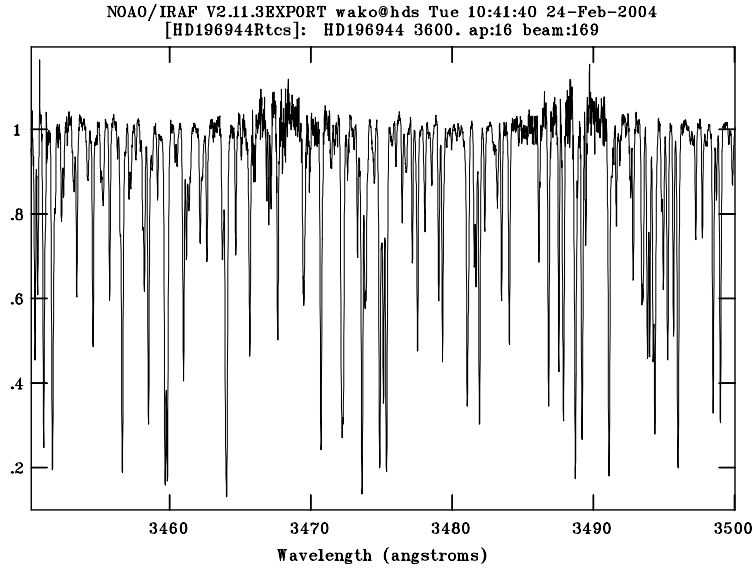


図 20: 規格化したスペクトルを単純に平均をとって一本化した場合。オーダの端に対応するところでデータの質が非常に悪い。

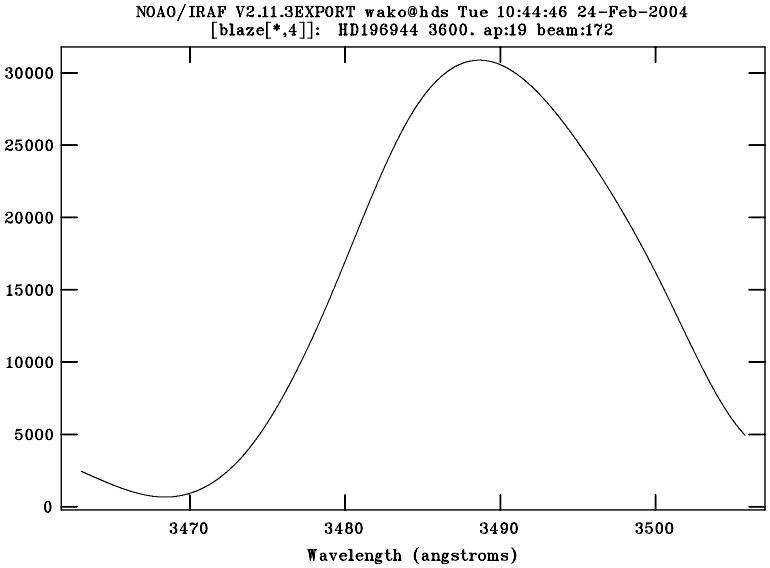


図 21: コンティニュームのスペクトル

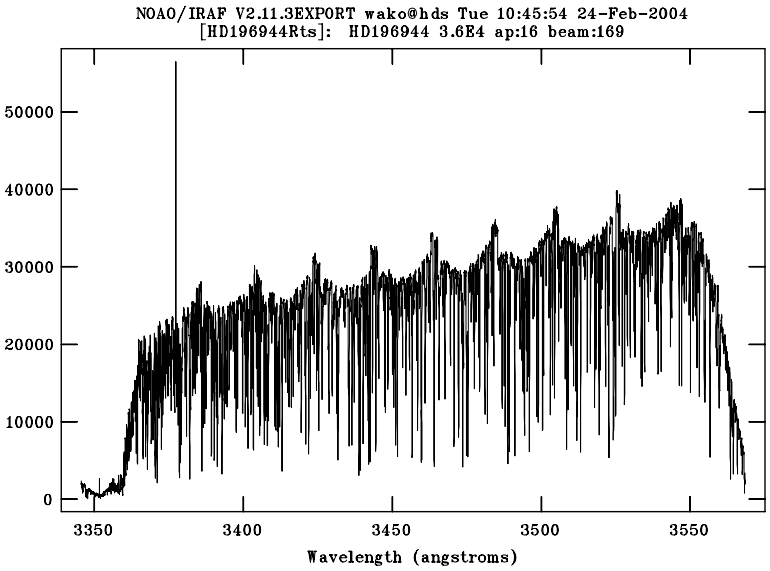


図 22: データの和をとることで一本化したスペクトル

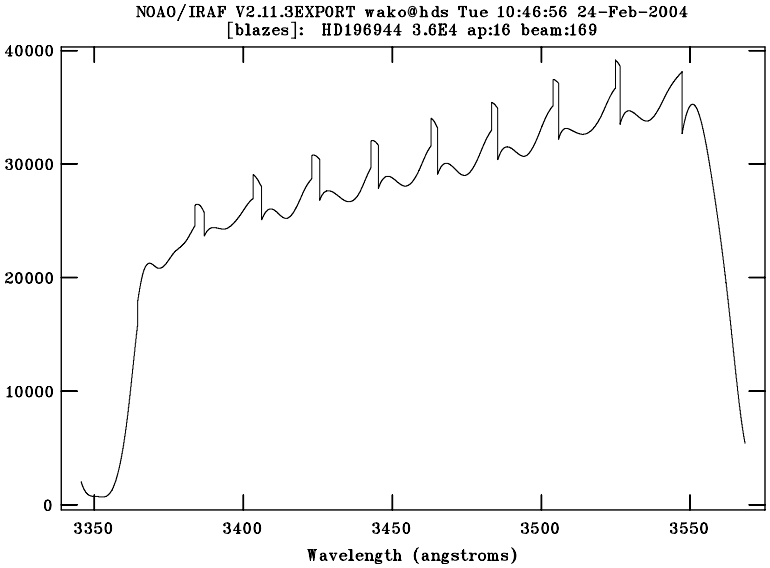


図 23: 図 22 と同じ方法で得られたコンチィニィームのスペクトル

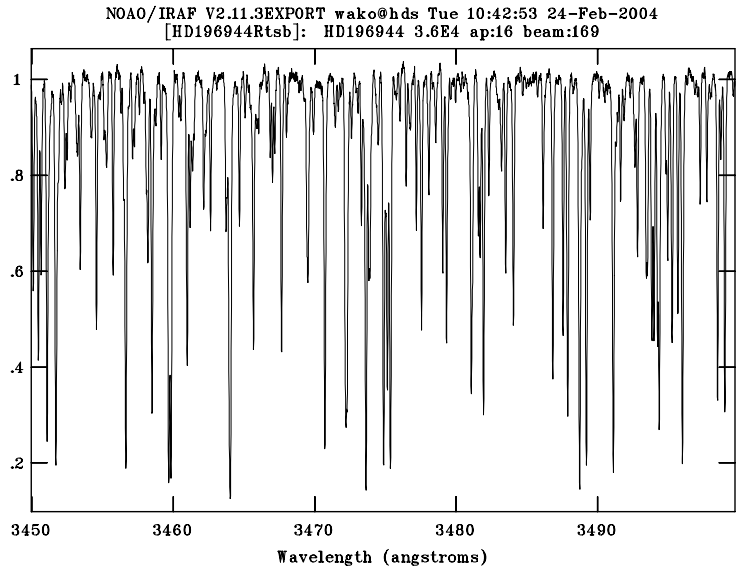


図 24: 図 22 のスペクトルを図 23 のコンティニュームで割って得られたスペクトル

15　その後の解析

以上で一次元スペクトルを得るところまではできました。その後の解析にも、IRAF は有用です。以下、順不同で紹介します。詳しくはそれぞれのタスクのヘルプをみてください。

15.1　データの統計量

データの統計量 (平均、標準偏差など) は、'imstatistics' というタスクで算出できます。

```
cl> imstat object
#               IMAGE      NPIX      MEAN    STDDEV       MIN       MAX
              OBJECT    92822    0.8684     10.98    -2594.     219.1
```

ファイル名 (上では 'object') については、統計をとる範囲を指定することもできます (object[1000:3000,*])。
このタスクは、S/N の測定などの用いることができます。

15.2　スペクトルの演算

一次元になったスペクトルデータの演算には、sarith や scombine が有用です。これらは、二次元データの場合の imarith や imcombine に対応するものですが、たとえば、scombine を使うと、波長範囲の異なるスペクトルを重ねることができるなど、スペクトルデータの処理に特有の機能ももっています。

15.3　スペクトル線の測定

星のスペクトルにみられる吸収線からは、さまざまな情報が引き出せます。吸収線の位置（波長）からは、その吸収線がつくられたところの視線速度が測られます。また、吸収線の幅は、そこの速度分散（熱運動などによる）を表していると考えられます。さらに、吸収線の強度（等価幅）からは、天体のモデルを介しての解析によって、吸収物質の量（柱密度）を見積もることができます。

スペクトル線の測定として、頻繁に用いられる標準的な方法は、一本の吸収線 (もしくは輝線) にたいして、ガウシアン曲線をフィットし、中心波長や線幅（半値幅)、等価幅 (もしくはフラッス) を測る、という方法です。前出の splot にはその機能が備わっていて、画面上、測定するスペクトル線の前後のコンティニュームの位置でそれぞれ 'k' を入力すると、ガウシアン曲線をフィットし、上述の値を計算してくれます（詳しくはヘルプを参照)。

なお、課題には、スペクトルの分解能の測定が含まれていますが、これも、コンパリソンデータにあらわれる弱い輝線の半値幅から求めるのが一般的です。

15.4　視線速度の測定

スペクトルにあらわれている吸収線を、実験室で測定されたスペクトル線と同定することができたら、視線速度を測定してみましょう。観測された吸収線の波長を $\lambda_{\rm obs}$、実験室で測定された波長を λ_0 とすると、みかけの視線速度 $v_{\rm ap}$ は、以下のようになります。

$$v_{\rm ap} = c \times \frac{\lambda_{\rm obs} - \lambda_0}{\lambda_0} \tag{1}$$

できれば複数の吸収線に対して測定し、精度の向上をはかります。

こうして測定された、みかけの視線速度を、地球運動の補正された太陽中心の視線速度 $v_{\rm her}$ になおします。これには、まず観測所の情報を noao.observatory で登録します。パラメータは以下のとおりです (観測所の位置は、理科年表などで確認してみてください)。

```
                                  I R A F
                     Image Reduction and Analysis Facility
PACKAGE = noao
   TASK = observatory

command =                    set  Command (set|list|images)
obsid   =                obspars  Observatory to set, list, or image default
images  =                         List of images
(verbose=                     no) Verbose output?

(observa=                     oao) Observatory identification
(name   =                     OAO) Observatory name
(longitu=          -133.5963889) Observatory longitude (degrees)
(latitud=            34.5738889) Observatory latitude (degrees)
(altitud=                  372.) Observatory altitude (meters)
(timezon=                   -9.) Observatory time zone
override=                        Observatory identification
(mode   =                     ql)
```

次に、noao.astutil.rvcorrect で、見かけの視線速度を太陽中心の値に変換します。[12] 以下の例では、$v_{\rm obs}$ は 12.34 km/s にしています。パラメータの observa は、 obspars にしておきます（上で指定した観測所の情報を使うということです)。観測年月日、世界時間（UT）は、データのヘッダから読み取ってください（Dataset3 については、ヘッダに値が書かれていないので、補遺の表の日本時間 (JST) を UT に直して使ってください)。天体の位置は、（ヘッダに書かれていることもありますが）各自で調べて（確認して）ください。

```
                                  I R A F
                     Image Reduction and Analysis Facility
PACKAGE = astutil
   TASK = rvcorrect

(files  =                       ) List of files containing observation data
(images =                       ) List of images containing observation data
(header =                    yes) Print header?
(input  =                     no) Print input data?
(imupdat=                     no) Update image header with corrections?

(epoch  =                  2000.) Epoch of observation coordinates (years)
(observa=                obspars) Observatory
(vsun   =                    20.) Solar velocity (km/s)
(ra_vsun=                    18.) Right ascension of solar velocity (hours)
(dec_vsu=                    16.) Declination of solar velocity (degrees)
(epoch_v=                  2000.) Epoch of solar coordinates (years)

(year   =                    1996) Year of observation
(month  =                       1) Month of observation (1-12)
```

[12] rvcorrect を用いて、スペクトルに対してあらかじめ地球運動の補正を行うこともできます。

```
(day    =                   18) Day of observation
(ut     =                18.969) UT of observation (hours)
(ra     =                8.7278) Right ascension of observation (hours)
(dec    =               -7.2336) Declination of observation (degrees)
(vobs   =                 12.34) Observed radial velocity
(hjd    =      2450101.2953037) Helocentric Julian Day (output)
(vhelio =      20.483996801402) Helocentric radial velocity (km/s) (output)
(vlsr   =      5.3777605335745) Local standard or rest radial velocity (km/s) (o
(mode   =                   ql)
```

結果は以下のように表示されますので、VHELIO の値を読み取ります。

```
# RVCORRECT: Observatory parameters for OAO
#       latitude = 34.5738889
#       longitude = -133.5963889
#       altitude = 372.
##   HJD          VOBS   VHELIO     VLSR   VDIURNAL   VLUNAR  VANNUAL   VSOLAR
2450101.29530    12.34    20.48     5.38     -0.268    0.007    8.404  -15.106
```

こうして得られた（太陽中心の）視線速度を、simbad[13] などのデータベースに載っているこれまでの研究と比較してみましょう。

15.5 アスキーデータへの書き出し

onedspec パッケージに、wspectext というタスクがあります。これによって 1 本化したスペクトルデータをアスキーデータに書き出すことができます。

```
ec> oned
      aidpars@       dopcor         reidentify     sensfunc       specplot
      autoidentify   fitprofs       rspectext      setairmass     specshift
      bplot          identify       sapertures     setjd          splot
      calibrate      lcalib         sarith         sfit           standard
      continuum      mkspec         sbands         sflip          telluric
      deredden       names          scombine       sinterp        wspectext
      dispcor        ndprep         scoords        skytweak
      disptrans      refspectra     scopy          slist

on> wspectext H4998bfs_ecwtcs H4998bfs.txt
```

できたファイルを見てみると、ヘッダに続けてスペクトルデータが「波長」「相対強度」で書き出されています。

[13] http://simbad.u-strasbg.fr/sim-fid.pl

```
BITPIX  =                    8  /  8-bit ASCII characters
NAXIS   =                    1  /  Number of Image Dimensions
NAXIS1  =               110420  /  Length of axis
ORIGIN  = 'NOAO-IRAF: WTEXTIMAGE'  /
IRAF-MAX=                   0.  /  Max image pixel (out of date)
IRAF-MIN=                   0.  /  Min image pixel (out of date)
IRAF-B/P=                   32  /  Image bits per pixel
IRAFTYPE= 'REAL FLOATING     '  /  Image datatype
OBJECT  = ' HD140283         '  /
FILENAME= 'H4998BFS_ECWTCS   '  /  IRAF filename
....

4110.51308358849  0.9839716
4110.52538877657  0.990428
4110.53769396465  0.9826592
4110.54999915273  0.978662
4110.56230434081  0.9812679
4110.57460952889  0.9803735
4110.58691471697  0.974153
4110.59921990505  0.9779771
4110.61152509312  0.9771149
4110.6238302812  0.9719001
4110.63613546928  0.9783365
4110.64844065736  0.9836422

....
```

16 IRAF を用いたデータ整約：注意事項と便利な機能

16.1 注意事項

陥りやすいミスをまとめました。「おかしいな？」と思ったら見てみてください。

1. **IRAF 起動:** IRAF を始めるには、'cl' コマンドを用いますが、これは必ず'login.cl' ファイルの置いてある場所で行います。

2. **表示画面はむやみに消さない：**スペクトルなどを表示する画面 (irafterm) を、ウインドウの機能を用いて消さないようにしましょう。消してしまうと IRAF を (場合によっては強制終了し、) 再起動する必要があります。

3. **2 つの CCD を混同しない：** HDS は二つの CCD をモザイクにして使っており、それに対応して一回の露出で二つのファイルが作られます。長波長側が奇数番、短波長側が偶数番のファイル名となっています。データ処理はそれぞれ独立に行います。バイアスの差引、フラットフィールディング、波長較正などでは、奇数番、偶数番を混同しないように注意しましょう。

4. **implot と splot** 二次元画像の断面図をみるには、implot を用います。一方、抽出したスペクトルを見るには splot が便利です。どちらも、表示範囲を変更したくなることはよくありますが、その方法はかなり違っています（後述）。混同しないようにしましょう。

5. **ディレクトリ database:** apall や ecident を行った結果、どのようにアパーチャを決めたか、どのように波長較正を行ったかという情報は、database というディレクトリに 'apXXXX', 'ecXXXX' というような名前で書き込まれます。このあたりの作業をすっかりやり直したいという場合には、apXXXX や ecXXXX のファイルを消去してから行う必要がある場合があります。

16.2 IRAF の便利な機能

1. **ヒストリー機能** 少し前に実行したタスクをそのまま、あるいは少し修正して実行したくなることはよくあります。そういった場合には、'e' を入力すると、ひとつ前に実行した内容が表示され、矢印キーでさらに前に遡ることができます。それを修正して実行することも可能です。

2. **一括処理** 同じ処理内容を、多数のファイルに対して実行したくなることがあります。例えば、天体データを同じフラットフレームで割るというような場合です。このときには、処理を行いたいデータ（ファイル名）をリストにしておき、リストのファイル名に@マークをつけてタスクを実行すると、一括処理を行うことができます。たとえば、フラット (flat.fit) で天体 (object) データを割る際に、input_list という名前のファイルをつくって

   ```
   object1
   object2
   object3
   ...
   ```

   のように書いておき、フラットで割った結果を書き出すファイルを

   ```
   object1f
   object2f
   object3f
   ...
   ```

   のように作って output_list という名前をつけたとすると、

   ```
   imarith @input_list / flat @output_list
   ```

   というように実行することでフラットで割る作業を一気に行うことができます。

3. **splot** 抽出したスペクトルを表示するタスクで、'splot H4998b_ec' のように使います。スペクトルを表示し、波長方向に拡大表示します。複数のオーダーのデータがある場合には、'shift + 9' と 'shift + 0' ('shift +8' と 'shift + 9')、つまり「丸括弧」を押すことで次の（あるいは前の）オーダのスペクトル表示に切り替えることができます。

   また、表示範囲を変えるためには、画面上で 'w' を入力し、「ウインドウモード」に入ります。そこで、カーソルを適当なところにおいて 'j' を入力すると、カーソルから右側だけが表示されます（左側が切られる）。'j' のかわりに'k' を入れると、左側だけが表示されます（右側が切られる）。同じく、' t'、'b' で、上、下が切られて表示されます。'a' を入力すると、全体が表示されます。（なお、一つ入力すると、ウインドウモードは終了しますので、再びウインドウモードに入るには改めて'w' を入れる必要があります。）

   画面表示をプリントアウトするには、画面上で':.snap' を入力します。':. snap epsf' とすれば、eps ファイルに書き出されます。

4. **implot** 二次元データの断面図を表示するときなどに使います。表示範囲を変更するには、':x 1000 2000' のようにすると横軸を 1000-2000 ピクセルについて表示し、:'y 0 5000' のようにすると縦軸を 0-5000 (カウント）について表示することになります。