すばる/MOIRCS 分光データ解析講習会資料

吉川　智裕
京都産業大学神山天文台

2010 年 12 月 14 日

目 次

第１章　はじめに

本テキストでは、MOIRCS の多天体分光 (MOS) データを IRAF (Image Reduction and Analysis Facility) および、その python インターフェイスである PyRAF を用いてデータの整約と解析を行う手順の一例を示しています。読者は基本的な UNIX コマンド、PyRAF もしくは IRAF の使い方、ヘルプの見方等はすでに習得していることを想定しています。

サンプルデータには、多天体分光によって複数の高赤方偏移の銀河の輝線観測を行ったデータを採用しています。従って、複数の天体のスペクトルが撮られているデータから目的の天体を切り出し、複数枚のフレームを重ね合わせ、検出された輝線の測定を行うことを目的とします。しかしながら、多くの手順は、一般に近赤外線の低分散 ($R \sim 500 - 2000$) スリット分光データ解析を行うときの参考になるものと期待します。

なお、本テキストで解説している手順については PyRAF 環境で開発された MCSMDP (MOIRCS MOS Data Pipelines) を使って自動化されています。巻末に MCSMDP を用いた自動リダクションの手順についてもまとめています。本テキストは、自動リダクションの利用者にとっても中身を理解する助けになることを期待しています。

本テキストは、「すばる秋の学校 2010」のテキストとして講師の吉川智裕（京都産業大学 神山天文台 専門員）が作成しました。サンプルデータは、すばる望遠鏡共同利用観測 S07A-083 (SMOKA にアーカイブ済み) のデータに加えて、同観測 PI の秋山正幸氏 (東北大学大学院理学研究科 天文学専攻 准教授) に観測に使用したマスクデザインファイル (MDP ファイル) を提供していただいたものを使用しています。データの一部は、Yoshikawa et al. 2010 [5] に出版されています。

本テキストの作成にあたり、「すばる秋の学校 2010」の世話人を務められた国立天文台ハワイ観測所、光赤外研究部、天文データセンターの皆様に感謝します。

吉川智裕
京都産業大学神山天文台専門員
2010 年 12 月 14 日

第2章　MOIRCS MOS分光データ

2.1 MOIRCS

MOIRCS (Multi-Object InfraRed Camera and Spectrograph) は、すばる望遠鏡のカセグレン焦点に取り付けられる観測装置で、近赤外線 (0.9–2.5 μm) の波長域での撮像および分光機能を持ちます。撮像装置としては、$YJHK_s$ の広帯域フィルタと各種の狭帯域フィルタを備え、$4' \times 7'$ の視野を観測することができます。一方、分光装置としては、グリズムおよび VPH グリズムを使った低分散 ($R \sim 700$)、中分散 ($R \sim 1500$)、高分散 ($R \sim 3000$) 分光観測を行うことができます。表 2.1 に MOIRCS の分光モードで利用可能なグリズムの分散と波長域についてまとめます。分光モードの時は、焦点面に複数のスリットを切ったスリットマスクを置き、約 40 天体程度の多天体分光観測を行うことができます。これは、すばる望遠鏡の可視の多天体分光装置、FOCAS (Faint Object Camera and Spectrograph) と同様の機能ですが、MOIRCS のスリットマスクは、特に 2 μm 以上の波長で顕著になる装置などからの熱輻射を抑えるために、薄いアルミの板に加工され、観測装置の他の部分と同様に 100 K 程度まで冷却されています。

表 2.1: MOIRCS で利用可能なグリズム [a]

グリズム	波長域 [μm]	波長分解能 [R] [b]	分散 [Å/pixel]	備考
$zJ500$	0.9–1.78	700	5.57	
$HK500$	1.3–2.5	640	7.72	H バンドの波長
		820		K バンドの波長
$R1300$	1.16–1.34	1500	1.91	J バンド、4 次光[c]
	1.45–1.80	1600	2.61	H バンド、3 次光[c]
	2.00–2.40	1500	3.88	K バンド、2 次光[c]
$VPH-J$	中心波長が 1.23	3050	0.96	波長シフトあり[d]
$VPH-H$	中心波長が 1.65	2940	1.31	波長シフトあり[d]

[a] 実際にプロポーザルを書くときはすばるの Web ページを良く読み、必要に応じてサポートサイエンティストに相談してください。

[b] スリット幅が 0.″5 の時。

[c] 一つのグリズムに J, H, K バンドそれぞれのフィルタをオーダーソートフィルタとして合わせ、4 次光、3 次光、2 次光を拾うようにできています。次数が上がるほど効率が下がりますので、J バンドの効率は K バンドの半分以下になります。

[d] 空間方向にどの位置にスリットを切るかによって、中心波長が変化します。変化量はグリズム、チャンネルによって異なります。詳細はすばるの Web ページを参照してください。

MOIRCS は二つの検出器アレイ、HAWAII-2 によってデータを取得します。HAWAII-2 は 2048 pixel× 2048 pixel のフォーマットを持つ HgCdTe の検出器です。MOIRCS の焦点面でのサンプリングレートは、撮像モード、分光モードとも 0.″117/pixel です。MOIRCS に入った光は、望遠鏡の焦点面で二つの視野に分割された後、二つの光学系でそれぞれの検出器に別々に結像されます。そこで、分光モードで視野の端の方の天体でもスペクトルを捉えられるように、中心から少し内側へずら

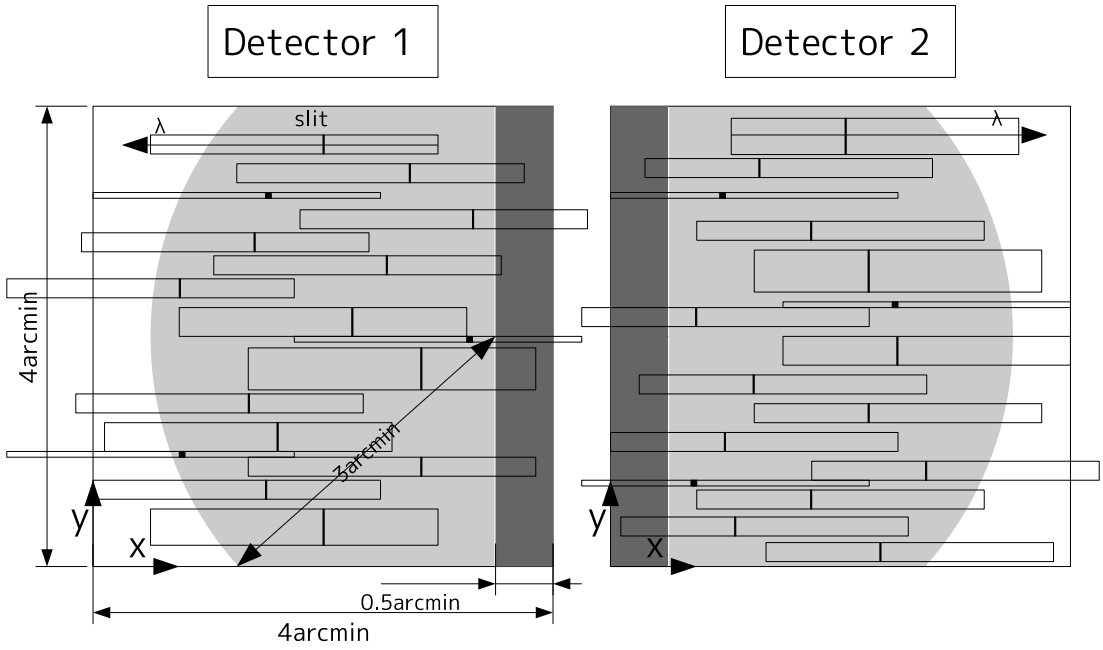

図 2.1: MOIRCS の二つの検出器と分光モード時のスペクトルの例。黒い長方形がスリット、白抜きの長方形がスペクトル。薄いグレーの 領域がスリットを切ることが可能な領域。スペクトルは x 軸の方向に分散され、検出器 1 では x 座標が増えるに従って波長が短くなり、検出器 2 ではその逆（VPH-J と VPH-H では分散の方向が逆になる）。白と薄いグレーの領域が撮像モードで観測可能な視野。分光モードのスペクトルは検出器全体に広がることができるが、スリットを切る位置によっては検出器からはみ出すこともある。

した位置に検出器が置かれています。そのため、撮像モードでは図 2.1 のように中心側の約 0.″5 (∼ 300 pixel) の領域には光が入らなくなっています。図の中では二つの検出器が離して描かれていますが、中心の領域が観測できないという意味ではありません。望遠鏡の焦点面で視野が二つに分割されているので、二つの検出器に写る $4' \times 3.''5$ の領域はほぼ連続しています。

2 つの検出器は、図 2.1 の通り検出器 1、検出器 2 と呼ばれます。観測装置のポジションアングル (PA) が 0 度の時に南に来るのが検出器 1、北に来るのが検出器 2 です。分光モードの時は、x 軸の方向に分散されるので、y 軸に沿ってスリットが置かれます。分散の方向は検出器によって異なります。検出器 1 は x 座標が増えるに従って波長が短くなり、検出器 2 はその逆です。検出器を並べたときに内側が短波長、外側が長波長となるような向きです。ただし、VPH-J と VPH-H グリズムは分散方向が逆になります。

MOIRCS の観測装置に付いての詳細は装置論文 (Suzuki et al. 2008 [4]) を、利用可能な観測モード等の詳細はすばる望遠鏡の Web ページ (http://naoj.org/Observing/Instruments/MOIRCS/index.html) を参照してください。

他のすばる望遠鏡の観測装置と同様に、MOIRCS のデータは FITS (Flexible Image Transport System) データ形式のファイルに格納されます。検出器毎に 1 つずつデータファイルを生成するため、1 回のデータ取得に対して 2 つの FITS ファイルが生成されます。データファイルには、すばる望遠鏡の制御システムによって一意のフレーム ID が付けられ、「(フレーム ID).fits」という名前のファイル名になります。フレーム ID は、例えば MCSA00057429 のようにアルファベットと数字から成ります。「MCS」はデータが MOIRCS によって取得されたこと、「A」はデータが観測装置から直接生成された生データであることを意味します。その後ろの 8 桁の数字がデータ毎に順番に振られる通し番号になっており、MOIRCS のファーストライトから今夜の観測まで、同じ数字を持つデータはありません。また、原則、検出器 1 から得られたデータは奇数、検出器 2 から得られたデータは偶数になるように番号が振られます。

FITS ファイルには、各ピクセルで検出された観測データの他に、観測がどのような条件で行われたどのようなデータか、といったデータ（メタデータ）を格納するための FITS ヘッダが入っています。FITS ヘッダはキーワード、値、注釈の 3 つで構成されるカードイメージの並びになっています。MOIRCS の場合、観測モード（撮像、ロングスリット分光、多天体分光）によってどのようなキーワードのカードイメージを FITS ヘッダに含めるかが決まっています。例えば、先ほどのフレーム ID は観測モードに拘らずキーワード「FRAMEID」というカードイメージに格納されており、仮にファイル名を変更してしまってもフレーム ID が何であったかがわかるようになっています。MOIRCS の FITS ヘッダに含まれる情報のうち、特に MOS モードに関係する重要なものを表 2.2 に示します。

FITS ファイルは天文分野の研究者、アマチュアの間で広く使われるデータ形式のため、読み書きのためのソフトウエアも数多く用意されています。本テキストでは、IRAF/PyRAF を用いた FITS ファイルの読み書きの仕方について後の章で説明します。FITS データ形式およびそれを扱うソフトウエア、及びすばる望遠鏡のデータに付加される FITS ヘッダの詳細に付いては、国立天文台天文データセンターが発行する FITS の手引きを参照してください。FITS の手引きはオンライン (http://www.fukuoka-edu.ac.jp/~kanamitu/fits/index.html) でも入手可能です。

表 2.2: MOIRCS の主要な FITS ヘッダ

キーワード	値の例	備考
FRAMEID	MCSA00057430	フレーム ID。生データのファイル名と同じ。
EXP-ID	MCSA00057429	露出 ID。同時にデータを撮った検出器 1 のフレーム ID と同じ。
DET-ID	2	検出器の ID。1 か 2。
OBS-MOD	SPEC_MOS	観測モード。撮像は IMAG、ロングスリット分光は SPEC。
DATA-TYP	OBJECT	データタイプ。他に、DARK, DOMEFLAT, INSTFLAT など。
OBJECT	CDFN_MASK01	観測のときに指定したターゲット名。
SLIT	CDFN1	スリットマスク名。サポートサイエンティストが命名する。
DISPERSR	HK500	使用したグリズムの名前
EXPTIME	900.050	露出時間。
K_DITWID	3.000	観測に使用したディザ幅。
K_DITCNT	1	ディザの何点目のデータか。分光は 1 か 2。
DATE-OBS	2010-12-14	観測を開始した時の UT での日付。
HST-STR	22:25:30.234	観測を開始したハワイ時間での時刻。
UT-STR	08:25:30.234	観測を開始した UT 時刻。

2.2 近赤外線分光観測

近赤外線分光観測[1]は、可視光の分光観測と基本的には同じですので FOCAS 等、可視光の低分散分光観測のデータ解析に使われる方法等をほとんどそのまま適用することができます。しかしながら、近赤外線特有の違いがいくつかあります。まずは、MOIRCS を使って行う近赤外線分光観測の特徴について、他のすばるの観測装置との違いを示しながら説明したいと思います。他の観測装置のテキストも参照しながら比較して理解してもらえればと思います。

FOCAS や Suprime-Cam 等、可視の観測に使われるシリコンの CCD (Charge Coupled Device) は 1 μm 以上の波長では感度が無くなるため、より長い波長の光に感度のある半導体結晶を用います。MOIRCS は、HAWAII-2 と呼ばれる HgCdTe 結晶の検出器を使用しています。一般に良く使われるシリコンの結晶と比べると製造技術が進んでおらず、特に MOIRCS で使用する HAWAII-2 は初期の大フォーマット近赤外線検出器であるため、大きな感度ムラや、感度の無いピクセル（バッドピクセル）等、CCD と比べて素性が良くない部分もあり、チップ毎の個性の違いも大きいです。また、シリコンとは異なりこのような結晶では電荷転送が行えないので、マルチプレクサを使ってピクセル毎に電荷を読み出す方法を使用します。このため、露出の前後に読み出しを行うダブルサンプリングの手法を使い機械シャッター無しで露出時間を決められるのも特徴の一つです。

近赤外線の波長域では、地球の大気中の水蒸気 (H_2O) や二酸化炭素 (CO_2) によって宇宙からの光が吸収されてしまいます。そのため、観測可能な波長が「大気の窓」と呼ばれる波長帯に限られます。図 2.2 にマウナケア山頂での大気の透過率と MOIRCS で使われる広帯域フィルタの透

[1] 本テキストでは特に断りが無い限り、近赤外線とは MOIRCS で観測可能な 0.9–2.5 μm の波長域を指すことにします。

過曲線を示します。このように近赤外線撮像観測用のフィルタは大気の窓に合わせて設計されますが、分光観測の際も同様に大気の窓に気をつける必要があります。例えば、$HK500$ グリズムは 1.3–2.5 μm の波長を同時に観測することができますが、H バンドと K バンドの間である 1.9 μm 前後の波長のスペクトルは大気に吸収されて地上からはほとんど観測することができません。

図 2.2 には MOIRCS で観測された背景光も一緒にプロットされています。背景光の主要な源は大気中の OH や O_3 からの夜光輝線です。大気の吸収同様、観測したいスペクトルが強い夜光輝線と同じ波長になってしまうと背景ノイズが大きすぎて観測できなくなってしまいます。FMOS とは異なり、OHS (OH Suppressor) のような夜光輝線を抑える機構がないため、データ整約の際にソフトウエア処理で夜光輝線を除いてやる必要があります。

一方、OH 夜光輝線の波長はよく調べられているため、近赤外線の分光観測では OH 夜光輝線を波長較正に使うことができます。そのため、可視の観測装置のように波長較正用のランプを別に撮る必要がほとんどありません。そこで、夜光のカウントが検出器上で saturate しない程度に露出時間を調整する必要があります。低分散分光グリズムで典型的な露出時間は 10–15 分程度になります。ただし、夜光の強度は時間変動が大きく、後述する A-B 引きである程度背景光を引くためにはもう少し短い露出時間が良いかもしれません。

輝線の影響が強い一方で、背景光のうち連続光は比較的小さいことがわかります。IRCS や COMICS のように 3 μm よりも長い波長を観測する装置の場合は、大気や望遠鏡からの熱輻射が強くなりますが、MOIRCS の場合は装置内で瞳に冷たい絞り（cold stop）を置くことによって 2.5 μm 以下ならば熱背景の影響を抑えることが可能です。しかしながら、低分散グリズムの観測では残った熱背景や分解しきれない夜光輝線の影響で、バックグラウンドリミットになることが経験的に知られています。

可視の観測に比べると近赤外線の観測は歴史も浅く、フラックス較正に使うための標準星のカタログも充分に整備されていません。特に、8m クラスの望遠鏡で観測可能な暗い天体の分光標準星のカタログは今のところありません。これについて、どのようにフラックス較正を行うかは後の章で議論します。

近赤外線観測や、そのための検出器についての詳細は、Ian McLean の”Electronic Imaging in Astronomy; Detectors and Instrumentation” [2] に詳しいです。興味のある方は参照してみてください。

2.3　MOIRCS MOS 観測

MOIRCS の MOS 観測の手順は、基本的には可視のスリットマスク多天体分光装置である FOCAS と同じです。観測者は、観測の前に視野のどこにスリットを切るのか、等を決め（マスクデザイン）、観測所がそれに従ってスリットマスクを作成します。そして、観測の際にはデザインした通りの視野に望遠鏡を向け、スリットマスクを焦点面に導入して観測を開始します。この節では、MOIRCS MOS 観測の準備と観測の流れについて、データ整約をする上で知っておいた方が良いことをまとめておきます。

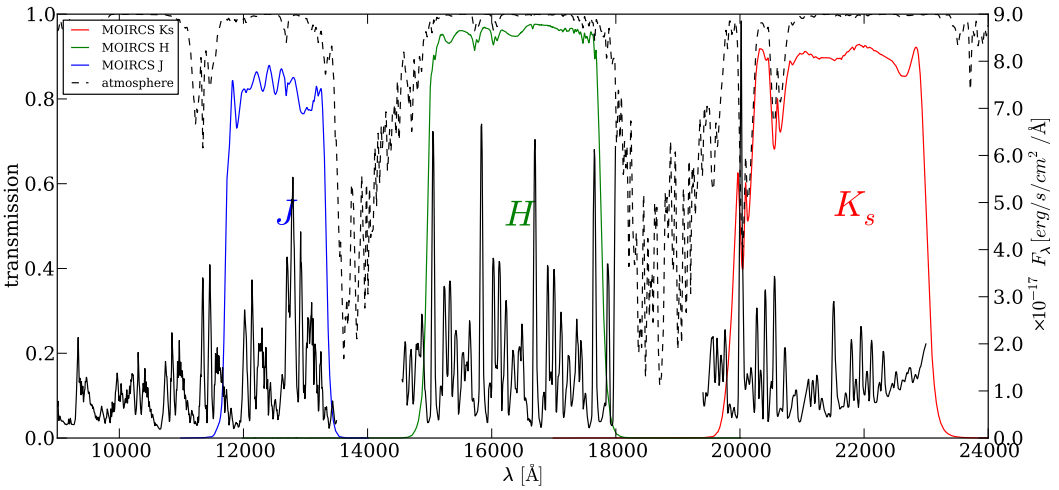

図 2.2: 左の軸: マウナケア山頂での大気の透過率（ダッシュ線）と、MOIRCS のフィルタの透過曲線（J: 青の実線、H: 緑の実線、K_s: 赤の実線）。右の軸: MOIRCS で観測した背景光（黒の実線、大気吸収が強い部分は省略）。

2.3.1 観測準備

スリットマスクをデザインするためには、目標となる領域付近について、目印となる天体が十分に検出可能な撮像データ（プレイメージ）が必要になります。視野内のスリットが正確に目標の天体に乗るために、プレイメージのデータは、すばる望遠鏡の焦点面での像を正確に再現する必要があります。視野のディストーション補正やピクセルスケールの不定性などを抑えるためには、MOIRCS 自身の撮像モードで取得したデータを使用するのが観測を成功させるためには最も確実な方法です。

共同利用で MOIRCS の MOS 観測に採択されると、プレイメージングの視野の座標とポジションアングル (PA) を決めるようにサポートサイエンティストから PI の元に連絡が来ます。PI は、座標と PA を指定し、サポートサイエンティストはそれに従って他の共同利用観測のすき間時間にプレイメージングのデータを取得し、マスクデザインを行えるように簡易処理して PI に送ります[2]。

マスクデザインは、mdp ファイルと呼ばれるテキストファイルを作成します。このテキストファイルはスリットの座標やスリットの大きさ、角度などが記述されたファイルで、FOCAS で使われる物と同じフォーマットです。mdp ファイルの作成を支援する wmdp_moircs というソフトがハワイ観測所で公開されていますのでそれを利用するのが便利です。生成した MDP ファイルは、wmdp_moircs を使って観測所のスリットを切る装置（レーザーカッター）が読み取れるフォーマットのファイル（sbr ファイル）に変換されます。この時、冷却によってスリットマスクが縮む量が

[2]プレイメージングで他の人の観測時間をもらうことになるため、自分の観測の時には次の MOS 観測の人のためのプレイメージングの時間を提供することになっています

経験的な値から考慮されます。

観測者が作成したマスクデザインの情報を観測所に送ると、観測所は、アルミの板の上に観測者がデザインした通りにスリットマスクを作成します。そして、観測の日に 100K に冷却された状態で使えるように、事前に観測装置内のスロットにインストールしておきます。

2.3.2　観測の流れ

マスクデザインをした通りに全ての天体をスリットに乗せるためには、望遠鏡をデザイン通りの方向に正確に向ける必要があります。そのため、全てのスリットマスクには導入用の星（点源）を 6 個以上入れるための丸い穴をデザインしておきます。視野導入の流れを図 2.3 にまとめます。大抵の場合、この作業は良く慣れたサポートサイエンティストや観測所の装置オペレータが行います。

導入用の星が穴の真ん中に入るとグリズムを入れて分光観測が開始されます。分光観測の場合は、特に長時間に渡って望遠鏡が目的の領域に正確に向いている必要があります。そのため、ガイド星を使ったオートガイディングを行います。また、露出毎に望遠鏡をスリットに沿って天体がスリットから外れない程度に上下に動かします（ディザリング）。これは、前述のように夜光の輝線が時間変動するため、近い時間のデータ同士で引き算をして夜光を引くためです。検出器上での天体の位置を動かすことによって、画像同士の引き算で夜光を引くことができます。撮像観測と同様に、検出器上の複数の位置に同じ天体を置くことによって、バッドピクセルや感度ムラなどの系統的な誤差を軽減する目的もあります。ただし、分光観測の場合は、通常は 2 点しかディザを取らないため、撮像観測に比べると効果は薄いです[3]。

分光観測の場合は特に長時間に渡るため、オートガイドをしていても望遠鏡や観測装置のたわみでマスクから天体が外れてきてしまうことがあります。1–2 時間に一回程度はグリズムを外して撮像データを取得し、導入穴の真ん中に星が来るように望遠鏡のポインティングを修正して観測を続けます。

2.3.3　較正用データ

観測の際に取得しておくべき較正用データは以下の通りです。

ダーク 背景のノイズの方がダークに比べてはるかに支配的なため、通常はデータ整約には使用しませんが、観測開始前に検出器の状態確認のために取得します。

フラット 明け方か夕方に使用するスリットマスクを望遠鏡の焦点面に導入してドームフラットを撮ります。検出器の感度ムラだけでなく、スリット加工時の幅のエラー等を補正する上で重要です。ドームフラットを撮る時とターゲットを観測するときでは、マスクの入れ直しや望遠鏡の姿勢の違いによる 6pixel 程度のマスク位置のずれが起きるので、それが気になる場合はターゲットに向けた状態でフラットプローブを焦点に入れてフラットを取る方法もあります（非常に時間が掛かるので最近は使われません）。

[3] 4 点取るモードも用意されていますが、一回の観測シーケンスの時間が非常に長くなるのを避けるためにほとんど使用されません

波長較正 明け方か夕方に比較光源（コンパリゾンランプ）を使って波長較正用のデータを取得する場合があります。前述の通り OH 夜光を使って波長較正をすることが可能なのでほとんど使われません。OH 夜光を使った方が望遠鏡の姿勢の違いによるスリットのずれが起きないので正確です。

標準星 A0V 型星のように、吸収線が少なくモデル化しやすい星を観測します。可視の分光標準星のようなカタログが整備されていないため、モデルを使ってフラックス較正をする必要があります。明け方か夕方の薄明の時間帯に取得しますが、大気の吸収量の変化が気になるときは夜半頃にも取得するのが確実です。時間省略のため、ターゲット用の MOS マスクの上でどれかひとつ適当なスリットを選んで分光します。

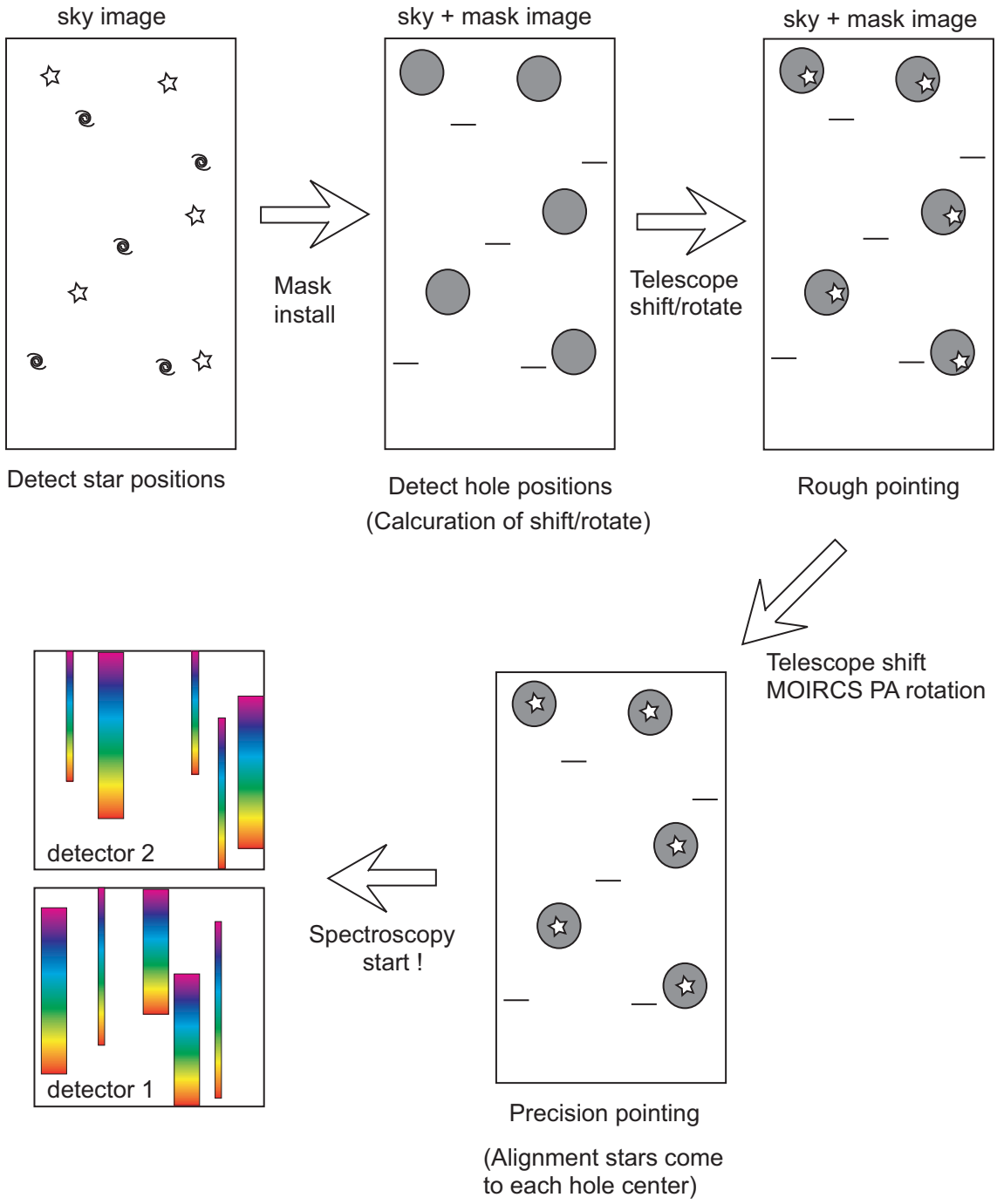

図 2.3: MOS 観測での視野導入の手順（東谷 2006, 東北大学博士論文）

第3章　データ整約概要

3.1　MOIRCS MOS データ整約の流れ

分光データの整約手順は、解析やデータの内容、目的等によって様々な方法があります。ここでの MOS データ解析は、非常に遠方の銀河にある電離領域からの輝線 (Hα) を検出し、その波長から赤方偏移を求め、Hα フラックスを測定する、ということを目的とします。そのためには、複数の枚数のフレームを足し合わせて 2 次元スペクトルの S/N を上げなくてはなりません。そこで、まずは x 軸を波長、y 軸を空間方向に対応させてあり、データの単位が specific flux (F_λ [erg s^{-1} cm^{-2} Å^{-1}]) となるような二次元スペクトルに整約します。その後で、二次元スペクトルから輝線を探して測定を行います。ここでは、観測データを二次元スペクトルへ整約するのにどのような作業を行うのかの流れを説明します。次章で、どのようにデータ整約を進めるかを IRAF の操作を追いかけながら実習します。

MOS 分光データの整約は基本的にはシングルスリット分光装置のスリット分光データの解析と同じです。ただし、一枚のフレームに複数のスリットのデータが乗っているため、一枚のフレーム全体に共通に行う手続き（共通処理）と、個別のスリットに切り出してから行う手続き（個別処理）の二つに分かれます。データ解析の流れを図 3.1 に示します。

共通処理は以下のような流れになります。

バッドピクセル・宇宙線処理 生データを見ると夜光の輝線が良く見えますが、それ以外にカウントの非常に高いピクセルがいくつかあることがわかります。これらのピクセルはそのままにしておくと足し合わせの時にノイズとなってしまいますので、先につぶしておく必要があります。原因は、検出器に固有のバッドピクセル（リニアリティが特に悪いもの、常にゼロカウントのもの、常に高いカウントのもの）と、フレーム毎に固有の宇宙線によるカウントです。フレームごとの宇宙線を検出し、既知のバットピクセルのマップと合わせ、これらを近くのピクセルの値で補完します。

A-B スカイ引き 近赤外線の背景光の主である夜光輝線は時間によって変動するため、近くの時間で撮ったフレーム同士で引き算して夜光を引くということはすでに述べました。ここで、同じディザセット内で撮ったデータ同士を引き算し、大雑把に夜光を引き去ります。

フラットフィールド 装置の光学系や検出器等に起因する感度のムラや、スリット加工時の幅エラーを補正します。較正用データの節で述べた通り、ドームフラットやフラットプローブなど、一様な光源を撮ったフラットフレームを作成し、その画像で割り算します。

ゆがみ補正 一般的に、望遠鏡、装置の光学系を通って検出器上にできる画像にはゆがみが生じます。近赤外線分光データの場合、夜光輝線をまっすぐにするという工程でゆがみが補正されますが、撮像用のゆがみ補正データベースが用意されていますので、先にある程度ゆがみを補正しておきます。

共通処理が終わると、スペクトルを天体毎に個別に切り出します。切り出したスペクトルに対して以下のような個別処理を行います。

波長較正 検出器上の座標と波長の関係を求め、波長が x 座標の一次関数になり、y 座標がスリット方向の空間座標に対応するように画像を変換します。OH 夜光輝線はあらかじめ波長がわかっていますので、これを利用します。

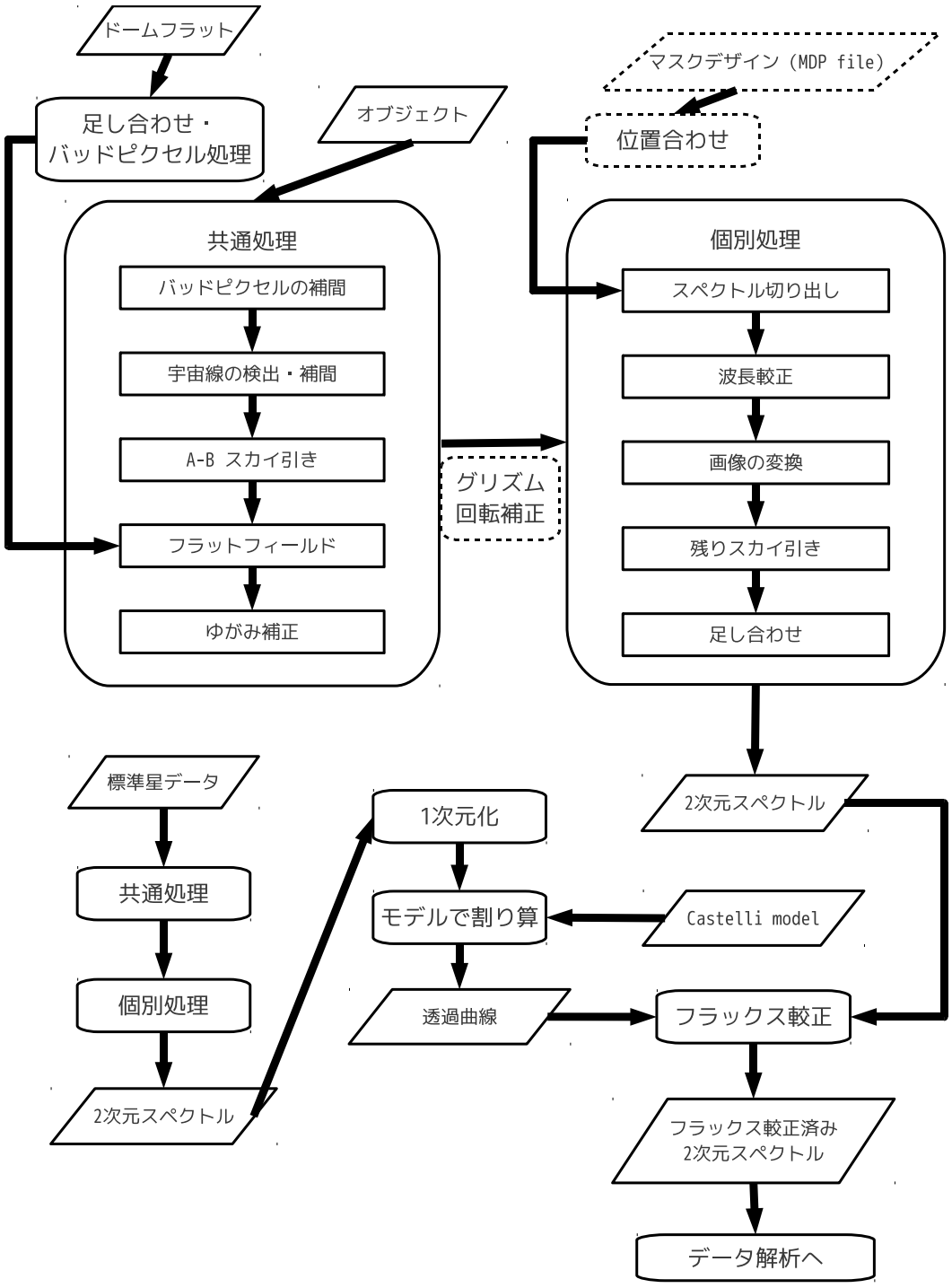

図 3.1: MOIRCS MOS データ整約の流れ

スカイ引き A-B スカイ引きで大雑把に背景光が引かれていますが、画像を見るとまだ夜光が残っていることがわかります。OH 夜光の変動によって A-B スカイ引きで引ききれていない成分です。そこで、空間座標方向に残りの背景光をフィットして引き算をします。

足し合わせ スカイ引きまでの処理を同じフィールドを観測した全てのフレームに行った後で、全てのフレームを足し合わせます。

フラックス較正 標準星のデータを使って 2 次元スペクトル上のカウントを波長当たりのフラックス (F_λ [erg sec^{-1} cm^{-2} Å^{-1}]) に直します。

3.2 MCSMDP

MCSMDP (MOIRCS MOS Data Pipelines) は MOIRCS ビルダの一人である京都産業大学神山天文台の吉川智裕が開発を行っている MOIRCS MOS データ整約用の PyRAF スクリプトパッケージです。京都産業大学のサーバー上[1]、もしくはハワイ観測所 Web ページのリンクから自由にダウンロードして研究に利用することができます。基本的に個人の研究用途に開発をしたものを公開しておりますので、必ずしも全てのデータに適切な処理ができるとは限りませんが、ある程度は一般のデータで使われることを想定して作ってあります。もしご自身の研究で使ってみて気づいた不具合や改善案などありましたらご連絡頂けると幸いです。

3.2.1 IRAF と PyRAF

IRAF (Image Reduction and Analysis Facility) は天文データの整約・解析のためにアメリカの NOAO (National Optical Astronomical Observatories) が開発したソフトウエアです。NOAO のサーバー[2]から無料でダウンロードして使用することができます。IRAF の対話的コマンドインターフェイス (cl と呼びます) 上で動作する多くのコマンド群 (タスクと呼びます) が用意されており、NOAO の観測装置だけでなく、世界中の多くの観測所で IRAF を利用した解析スクリプトが開発されています。

PyRAF は IRAF のタスクを Python という汎用のスクリプト言語上で動くようにしたコマンド言語で、NASA の STScI (Space Telescope Science Institute) が開発し、無料で配布[3]しています。PyRAF は cl の代替となるユーザーフレンドリーなコマンドラインを提供し、IRAF でできることは基本的に PyRAF でも可能です。それだけでなく、PyRAF タスクは Python スクリプトとして記述することができるため、豊富な科学計算用の Python ライブラリを利用して IRAF の処理を容易に拡張することができます。

MCSMDP は多数の PyRAF タスクで構成されており、PyRAF のコマンドライン上で動作します。現在の MCSMDP の開発環境は IRAF version 2.14.1 (lnux 版)、PyRAF version 1.9 です[4]。特に PyRAF の更新頻度が高いので、なるべく最新の環境で動作するように開発を続けています。

[1] http://www.cc.kyoto-su.ac.jp/~tomohiro/MCSMDP/
[2] http://iraf.nao.ac.jp/iraf/web/
[3] http://www.stsci.edu/resources/software_hardware/pyraf
[4] 2010 年 11 月に IRAF version 2.15 がリリースされました。PyRAF での動作はまだ公式にサポートされていないようです。

開発環境は Ubuntu Linux 10.4 ですが、三鷹の解析環境を含めて IRAF/PyRAF が動く環境ならどこでも動くと思います。

3.2.2 必要なソフトウエア

MCSMDP の動作には、以下のソフトウエアが必要です。環境によっては、開発環境パッケージ (Ubuntu Linux なら *-dev パッケージ等) が必要になる場合があります。使用している環境のパッケージ管理システム (apt, yum 等) を利用して、適宜、インストールをしてください。

IRAF, Python stsci_python に必要とされるバージョンのものをインストールしてください。

stsci_python PyRAF を含む python モジュール群です。STScI の Web サイトからダウンロードしてインストールします。インストール方法は同 Web サイトを参照してください。IRAF, Python を含め、PyRAF の動作に必要とされるソフトウエアもインストールしてください。

ds9, xpa Harvard-Smithonian Center for Astrophysics の Web サイト[5]で配布されています。ximtools はサポートされていません。

RO Python Package ワシントン大学の Russel Owen 氏が開発、配布[6] している Python モジュールです。Python の setuptools がインストールされている場合は、easy_install を使ってインストールすることも可能です。

以下のソフトウエアはデータ整約には必要ありませんが、輝線の解析をするタスクを利用するときには必要になります。

matplotlib グラフ描画の python モジュールです。公式サイト[7]で配布されています。多くの Linux ディストリビューションにも含まれています。

scipy 科学計算ライブラリの python モジュールです。公式サイト[8]で配布されています。多くの Linux ディストリビューションにも含まれています。

ipython python をインタラクティブに使用するためのシェルを提供するプログラムです。PyRAF のフロントエンドとしても使用でき、PyRAF のコマンドラインよりも機能が豊富です。公式サイト[9]で配布されています。多くの Linux ディストリビューションにも含まれています。

R, rpy フリーの統計解析ソフトウエアとその python インターフェイスです。それぞれ、公式サイト[10][11] で配布されています。多くの Linux ディストリビューションにも含まれています。

[5] http://hea-www.harvard.edu/RD/ds9/
[6] http://www.astro.washington.edu/users/rowen/ROPackage/Overview.html
[7] http://matplotlib.sourceforge.net/
[8] http://www.scipy.org
[9] http://ipython.scipy.org/moin
[10] http://www.r-project.org/
[11] http://rpy.sourceforge.net/

3.2.3　インストール

まず、MCSMDP（スクリプト本体）と、MDPDB（較正用データベース）を MCSMDP のサイトからダウンロードし、任意のディレクトリに展開します[12]。ここでは、home ディレクトリ直下で展開しているとします。

```
$ wget http://www.cc.kyoto-su.ac.jp/~tomohiro/MCSMDP/MCSMDP_v1_0_1.tgz
$ wget http://www.cc.kyoto-su.ac.jp/~tomohiro/MCSMDP/MDPDB_v1_0_1.tgz
$ tar xvfz MCSMDP_v1_0_1.tgz
$ tar xvfz MDPDB_v1_0_1.tgz
```

MCSMDP のディレクトリに入り、setup.sh を実行します。

```
$ cd MCSMDP_v1_0_1
$ ./setup.sh
```

ログインシェルが bash, ksh, sh, zsh の場合[13]

ホームディレクトリの.bash_profile (bash のとき) または、.profile (ksh, sh, zsh のとき) ファイルの最後に以下を記述します。

```
source ${HOME}/MCSMDP_v1_0_1/etc/mcsmdp.sh
```

ログインシェルが tcsh, csh の場合

ホームディレクトリの.cshrc ファイルの最後に以下を記述します。

```
source ${HOME}/MCSMDP_v1_0_1/etc/mcsmdp.csh
```

3.2.4　解析の準備

作業用のディレクトリを準備し、mcsmdp を起動してみましょう。mcsmdp を初めて実行するディレクトリでは、実行するディレクトリで iraf の環境設定が行われます。mkiraf を実行するかを聞かれますので、リターンキーを押してください[14]。しばらく待つと、pyraf シェル (-->) が起動します。

```
$ mcsmdp
mkiraf? (yes):
...
-->
```

[12] 以下、“$” から始まる行は bash のプロンプトにコマンドを入力していることを意味します。

[13] ここでは、天文データセンターの解析環境の場合について説明します。利用する環境によって、適切な設定を行ってください。

[14] mkiraf については iraf のドキュメントを参照してください。自動で mkiraf を実行されるのを望まない場合に限り、“no” と答えてください

シェルが起動したら、mcsmdp と打ち込んで、mcsmdp タスクを読み込んでください[15]。

```
--> mcsmdp
...
-->
```

pyraf シェルは IRAF の cl 同様、対話的にコマンドを打ち込んで操作します。pyraf 自体の操作方法の説明は本テキストの目的から外れますので詳細は省きますが、ここでは次章からの実習を進める上で基本的な操作方法について簡単にまとめておきます。詳細は STScI が配布する “The PyRAF Tutrial”[16] などを参照してください。

本テキストでタスクを実行する場面では、そのタスクのパラメータリストを示しています。例えば、mdpdisplay のパラメータリストを表示するには以下のようにします。

```
--> lpar mdpdisplay
        file = "HK500_MODS11-0390fl.fits" Image (list) to be loaded
     (frame = 1)                  Display frame to be loaded
(multiframe = no)                 Increment frame number for image list?
     (scale = "linear")           scale type
     (smode = "zscale")           scale mode
        (z1 = INDEF)              minimum greylevel to be displayed
        (z2 = INDEF)              maximum greylevel to be displayed
      (cmap = "BB")               color map
    (invert = no)                 invert color map?
      (mode = "al")
```

まず、パラメータリストをテキストに示されたように編集します。編集するときは以下のようにします。

```
--> epar mdpdisplay
```

すると、図 3.2 のようなパラメータエディタが起動します。パラメータ名に括弧 “()” がついている各項目を、パラメータリストの通り編集します。iraf と同様、括弧がついているパラメータは隠しパラメータで pyraf プロンプトから対話的に実行する場合も明示的に値を与える必要はなく、epar で編集、保存された値が使われます。編集が終わったら、エディタの上部に並んでいるボタンから、“Save & Quit” をクリックすると、編集した値を保存して pyraf プロンプトに戻ります。

本テキストでは使用しませんが、他のボタンについてもまとめておきます。“Execute” は、エディタに入力されている値をそのまま使ってタスクを実行します。“Unlearn” は、編集されたパラメータをデフォルトの値に戻します。“Cancel” は編集した値を保存せずに破棄してエディタを終了します。右端の “Mdpdisplay Help” はタスクのヘルプを表示するボタンですが、残念ながら MCSMDP ではヘルプドキュメントを整備していないので何も表示されません。通常の iraf のタスクの場合はヘルプドキュメントが別窓で表示されます。

[15] iraf では、loginuser.cl に読み込みたいタスクを記述しておくと、実行時に自動的に読み込まれるのですが、どうやら pyraf のタスクでそれをやるとエラーを起こしてしまうようです。

[16] http://stsdas.stsci.edu/stsci_python_epydoc/docs/pyraf_tutorial.pdf

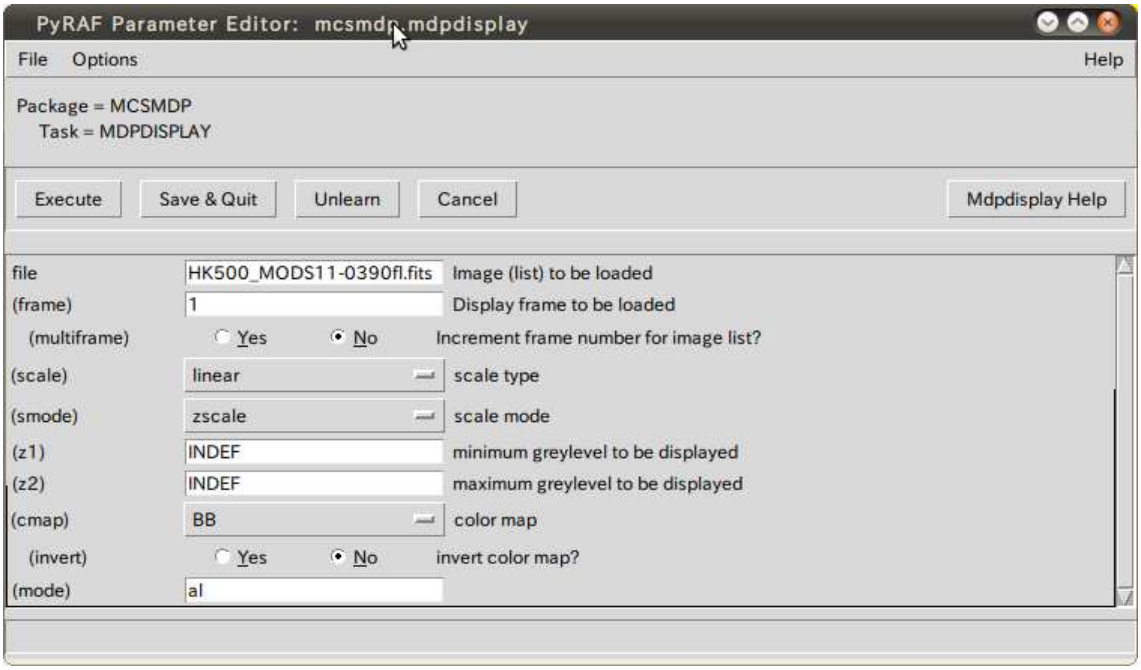

図 3.2: mdpdisplay のパラメータエディタ

タスクの実行は、pyraf プロンプト上で行います。先ほどのエディタで括弧の付いていなかったパラメータはその順番で並べる必要があります。また、括弧の付いていた隠しパラメータも、“パラメータ名=値” という形式でパラメータエディタに指定しなかった値を指定することもできます。この値は保存されず、1 回の実行限りです。

```
—> mdpdisplay MCSA00057147.fits frame=2
```

mcsmdp を終了する時は、“.exit” と入力してください。

```
—> .exit
$
```

参考・ipython モードでの起動

mcsmdp の実行時に--ipython オプションを付加すると、pyraf シェルの代わりに以下のような ipython シェルが起動します。

```
$ mcsmdp —ipython
...
In [1]:
```

ipython シェルモードを使用するときは、ファイル~/.ipython/ipythonrc-pyraf に以下の記述を追加してください。もしこのファイルが存在しないときは、上記コマンドで一度 ipython モードの mcsmdp を起動し、“Ctrl+D” で終了すると自動的にファイルが生成されます。

```
import_mod pylab
execfile pylabinit.py
```

ipython シェルでは matplotlib などの python のグラフ描画ツールを利用できるようになっています。pyraf シェルの方がオリジナルの IRAF の cl に近い操作になりますので、本テキストでは、主に pyraf シェルの方を使用します。

第4章　データ整約実習

4.1　データの準備

作業用のディレクトリを準備し、以下の要領でサンプルデータをコピーします[1]。

```
$ mkdir -p /mfs02a/tomohrys/MCSMDP_work # 作業ディレクトリ名は各自決める
$ cd /mfs02a/tomohrys/MCSMDP_work
$ cp /mfs01b/tomohrys/MCSMDP_sample/*.* .
```

サンプルデータには、拡張子が mdp, fits の 2 種類のファイルが含まれています。

拡張子が mdp のファイルは MDP ファイルと呼ばれるテキストファイルです。スリットマスクのデザイン情報が記述されており、観測者は観測の前にこのファイルを作成して観測所にスリットマスクの加工を依頼します。このファイルは、後でデータファイルから各スリットのスペクトルを切り出すときに使用します。

拡張子が fits のファイルが、FITS ファイルと呼ばれるデータファイルです。どのようなデータかを mcsmdp を起動して早速見てみましょう。hselect というタスクを使うのが便利です。

```
$ mcsmdp
--> mcsmdp
--> hselect *.fits $I,OBS_MOD,DATA_TYP,OBJECT,DISPERSR yes
MCSA00057015.fits SPEC_MOS OBJECT DOMEFLAT HK500
MCSA00057016.fits SPEC_MOS OBJECT DOMEFLAT HK500
MCSA00057017.fits SPEC_MOS OBJECT DOMEFLAT HK500
...
MCSA00057114.fits SPEC OBJECT M53735(A0V:J8.9:H8.9:K8.9) HK500
MCSA00057116.fits SPEC OBJECT M53735(A0V:J8.9:H8.9:K8.9) HK500
MCSA00057147.fits SPEC_MOS OBJECT CDFN_MASK02 HK500
MCSA00057148.fits SPEC_MOS OBJECT CDFN_MASK02 HK500
MCSA00057149.fits SPEC_MOS OBJECT CDFN_MASK02 HK500
...
```

OBS-MOD は観測モードで、今回はほとんどが多天体分光モード (SPEC_MOS) になっています。シングルスリット分光モード (SPEC) のデータは標準星のデータです。DATA-TYP は取得したデータのタイプで、DOMEFLAT となっているのがドームフラットのデータです。OBJECT は観測天体名、DISPERSR が使用した分散素子で今回は全て HK500 になっています。

データ整約の手続きに入る前に、これらのデータを分類してファイルのリストを作っておきます。ここでも hselect を使います[2]。

```
--> hselect MCSA*.fits $I
 "OBJECT = 'DOMEFLAT' & @'DET_ID' = 1" > flat1.lst
--> hselect MCSA*.fits $I
 "OBJECT = 'DOMEFLAT' & @'DET_ID' = 2" > flat2.lst
```

[1] サンプルデータは http://www.cc.kyoto-su.ac.jp/~tomohiro/MCSMDP/MCSMDP_sample.tbz にも置いてあります

[2] 紙面の都合で折り返していますが、実際は一行で打ち込んでください。以下も同様とします。

```
—> hselect MCSA*.fits $I
 "OBJECT = 'CDFN_MASK02' & @'DET-ID' = 1 & K_DITCNT = 1" > obj1a.lst
—> hselect MCSA*.fits $I
 "OBJECT = 'CDFN_MASK02' & @'DET-ID' = 1 & K_DITCNT = 2" > obj1b.lst
—> hselect MCSA*.fits $I
 "OBJECT = 'CDFN_MASK02' & @'DET-ID' = 2 & K_DITCNT = 1" > obj2a.lst
—> hselect MCSA*.fits $I
 "OBJECT = 'CDFN_MASK02' & @'DET-ID' = 2 & K_DITCNT = 2" > obj2b.lst
```

flat1, flat2 はそれぞれ検出器 1, 2 のドームフラットです。obj1a, obj1b は検出器 1 のオブジェクトデータで、それぞれ A 点、B 点です。obj2a, obj2b は同じく検出器 2 のオブジェクトデータです。標準星のデータは 2 枚だけなので、リストの作成は省略します。以下では、これらのファイルリストを使ってデータ整約を進めます。今回の実習に使う天体のスリットは検出器 1 の上にいますので、検出器 1 の方を例に進めますが、検出器 2 も同様ですので各自試してください。

4.2 共通処理

4.2.1 バッドピクセル、宇宙線の検出・補間

宇宙線の検出には craverage というタスクを使います。このタスクは、周りのピクセルの平均に対してカウントが高いピクセルを宇宙線として検出し、既知のバッドピクセルマスクに追加したマスクファイルを作成します。バッドピクセルマスクは 0 と 1 の値で埋められている画像ファイルで、1 の場所がバッドピクセルの場所を表しています。タスクの詳細はヘルプを参照してください。

まず、宇宙線・バッドピクセルマスクのリストを作成します。何でも構いませんが、ここでは “BPM” というディレクトリの下に元のデータファイルと同じ名前で拡張子を “.pl” に変えたファイルとします。ファイルリストの作成には sed という UNIX コマンドを使います。また、“BPM” ディレクトリを作成します。

```
—> !sed 's/\(.*\)\.fits/BPM\/\1\.pl/' obj1a.lst > bpm1a.lst
—> !sed 's/\(.*\)\.fits/BPM\/\1\.pl/' obj1b.lst > bpm1b.lst
—> mkdir BPM
```

既知のバッドピクセルマスクは観測所で用意されていますので、それを使用します。今回のデータ用のバッドピクセルマスクは MCSMDP のパッケージにも含まれており、検出器 1 用が mdpdb$bpm/nlbpm1_FF64r.fits、検出器 2 用が mdpdb$bpm/nlbpm2_FF64r.fits です。これを、上で用意したバッドピクセルマスクのファイル名になるようにコピーしておきます。craverage は入力されたバッドピクセルマスクを使い、既知のバッドピクセルを除いた領域から宇宙線の計算を行って元のバッドピクセルマスクに追加します。

```
—> imcopy mdpdb$bpm/nlbpm1_FF64r.fits,mdpdb$bpm/nlbpm1_FF64r.fits,
mdpdb$bpm/nlbpm1_FF64r.fits,mdpdb$bpm/nlbpm1_FF64r.fits @bpm1a.lst
—> imcopy mdpdb$bpm/nlbpm1_FF64r.fits,mdpdb$bpm/nlbpm1_FF64r.fits,
mdpdb$bpm/nlbpm1_FF64r.fits,mdpdb$bpm/nlbpm1_FF64r.fits @bpm1b.lst
```

epar で以下のように craverage のパラメータを編集します。

```
    input = ""      List of input images
   output = ""      List of output images
  (crmask = "")     List of output cosmic ray and object masks
 (average = "")     List of output block average filtered images
   (sigma = "")     List of output sigma images

    (navg = 15)     Block average box size
    (nrej = 100)    Number of high pixels to reject from the average
    (nbkg = 5)      Background annulus width
    (nsig = 10)     Box size for sigma calculation
    (var0 = 0.0)    Variance coefficient for DN^0 term
    (var1 = 0.0)    Variance coefficient for DN^1 term
    (var2 = 0.0)    Variance coefficient for DN^2 term

   (crval = 1)      Mask value for cosmic rays
  (lcrsig = 100.0) Low cosmic ray sigma outside object
  (hcrsig = 10.0)  High cosmic ray sigma outside object
  (crgrow = 0.0)    Cosmic ray grow radius

   (objval = 0)     Mask value for objects
  (lobjsig = 10.0)  Low object detection sigma
  (hobjsig = 5.0)   High object detection sigma
  (objgrow = 0.0)   Object grow radius
```

craverage を実行します。

```
--> craverage @obj1a.lst "" crmask=@bpm1a.lst
--> craverage @obj1b.lst "" crmask=@bpm1b.lst
```

次に、fixpix というタスクを使って、作成したバッドピクセルマスクに従ってバッドピクセル部分を補間して埋めます。今回の解析では、スペクトルは空間方向に潰してフラックスを測るので、波長方向の情報は変えないように、空間方向である Y 軸方向のみに沿って線形補完します。

先ほどと同様に出力ファイルリストを作成します。ここではファイル名の頭に “cr” を付けたものを宇宙線処理済みのファイルとします。

```
--> !sed 's/\(.*\)/cr\1/' obj1a.lst > crobj1a.lst
--> !sed 's/\(.*\)/cr\1/' obj1b.lst > crobj1b.lst
```

fixpix は入力ファイルを書き換えるので、このリストのファイル名のファイルにコピーしておきます。

```
--> imcopy @obj1a.lst  @crobj1a.lst
--> imcopy @obj1b.lst  @crobj1b.lst
```

epar で以下のように fixpix のパラメータを編集します。

```
      images =                  List of images to be fixed
       masks =                  List of bad pixel masks
   (linterp = "INDEF")          Mask values for line interpolation
   (cinterp = "1")              Mask values for column interpolation
   (verbose = no)               Verbose output?
    (pixels = no)               List pixels?
```

fixpix を実行します。

```
--> fixpix @crobj1a.lst @bpm1a.lst
--> fixpix @crobj1b.lst @bpm1b.lst
```

参考・MCSMDP での宇宙線・バッドピクセル処理

宇宙線・バッドピクセルマスクの処理は手順が複雑なので、ここでは簡略化した方法を説明しました。MCSMDP では、crrejection というタスクを用意して以下のような手順で処理をしています。

1. 生データから宇宙線を検出 (宇宙線 1)
2. A-B ペアの相方で割ったフレーム（A 点なら A/B）から宇宙線を検出（宇宙線 2）
3. 宇宙線 1, 宇宙線 2, バッドピクセルマスクのフレームで OR を取って合成した宇宙線・バッドピクセルマスクを作成する
4. 空間方向に補間
5. 検出器象限境界のピクセルを波長方向に補間

宇宙線 2 フレームを作るのは、近赤外線の分光データは夜光輝線のカウントが支配的なので生データだけでは宇宙線の検出が難しく、同程度の夜光輝線カウントを持ったフレームで割ることによって宇宙線を浮き立たせるためです。実際、検出される宇宙線の数が多少増えますが、まだ不十分です。宇宙線の検出のパラメータのチューニング、手法の改良は残された課題の一つです。

検出器象限境界には 1 ピクセル幅の隙間がありますが、縦方向の隙間は今の方法では補間することができません。そこで、このピクセルに限って波長方向に補間しています。

crrejection を使うときは、以下のようにパラメータを設定します。

```
 inimage1 = ""          input frame at A position
 inimage2 = ""          input frame at B position
outimage1 = ""          output frame at A position
outimage2 = ""          output frame at B position
      bpm = ""          badpixel mask
    (navg = 15)         Block average box size
```

```
    (nrej = 100)      Number of high pixels to reject from the average
    (nbkg = 5)        Background annulus width
    (nsig = 10)       Box size for sigma calculation
    (var0 = 0.0)      Variance coefficient for DN^0 term
    (var1 = 0.0)      Variance coefficient for DN^1 term
    (var2 = 0.0)      Variance coefficient for DN^2 term
  (lcrsig = 100.0)    Low cosmic ray sigma outside object
  (hcrsig = 10.0)     High cosmic ray sigma outside object
  (crgrow = 0.0)      Cosmic ray grow radius
  (bpmdir = "BPM")    directory name to store bad pixel masks
```

crrejection を実行します (ここでは、今作ったファイルを消さないように “cr2*” というファイルに出力し、バッドピクセルマスクは “BPM2” というディレクトリに入るようにしています)。

```
--> !sed 's/\(.*\)/cr2\1/' obj1a.lst > cr2obj1a.lst
--> !sed 's/\(.*\)/cr2\1/' obj1b.lst > cr2obj1b.lst
--> crrejection @obj1a.lst @obj1b.lst @cr2obj1a.lst @cr2obj1b.lst
 mdpdb$bpm/nlbpm1_FF64r.fits bpmdir="BPM2"
```

4.2.2　A-B スカイ引き

次に、A 点と B 点のペアでスカイを引き算します。imarith というタスクを使います。結果のファイルは先頭に “ab” を付けることにします。

```
--> !sed 's/\(.*\)/ab\1/' crobj1a.lst > abobj1a.lst
--> imarith @crobj1a.lst - @crobj1b.lst @abobj1a.lst
```

4.2.3　ドームフラットの作成とフラットフィールド

まず、ドームフラットのデータからドームフラットを作成します。imcombine のパラメータを以下のように編集します。

```
      input =           List of images to combine
     output =           List of output images
   (headers = "")       List of header files (optional)
   (bpmasks = "")       List of bad pixel masks (optional)
  (rejmasks = "")       List of rejection masks (optional)
 (nrejmasks = "")       List of number rejected masks (optional)
  (expmasks = "")       List of exposure masks (optional)
    (sigmas = "")       List of sigma images (optional)
     (imcmb = "$I")     Keyword for IMCMB keywords
```

```
   (logfile = "STDOUT")   Log file

   (combine = "median")   Type of combine operation
    (reject = "sigclip")  Type of rejection
   (project = no)         Project highest dimension of input images?
   (outtype = "real")     Output image pixel datatype
 (outlimits = "")         Output limits (x1 x2 y1 y2 ...)
   (offsets = "none")     Input image offsets
  (masktype = "none")     Mask type
 (maskvalue = "0")        Mask value
     (blank = 0.0)        Value if there are no pixels

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

(lthreshold = INDEF)      Lower threshold
(hthreshold = 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 = 3.0)        Lower sigma clipping factor
    (hsigma = 3.0)        Upper sigma clipping factor
   (rdnoise = "0.")       ccdclip: CCD readout noise (electrons)
      (gain = "1.")       ccdclip: CCD gain (electrons/DN)
    (snoise = "0.")       ccdclip: Sensitivity noise (fraction)
  (sigscale = 0.1)        Tolerance for sigma clipping scaling corrections
     (pclip = -0.5)       pclip: Percentile clipping parameter
      (grow = 0.0)        Radius (pixels) for neighbor rejection
```

imcombine を実行します。

```
--> imcombine @flat1.lst HK500_CDFN2_Domeflat1.fits
```

このままでもフラットフレームとして使えますが、カウントが大きいためこのフラットで割るとオブジェクトのデータのカウントが小さくなってしまいます。そのため、大体1に近いカウントになるように割り算をしておきます。

```
--> imarith HK500_CDFN2_Domeflat1.fits / 10000.
 HK500_CDFN2_Domeflat1.fits
```

次に、オブジェクトフレームと同じように fixpix でバッドピクセルの処理をしておきます。パラメータは先ほどと同じです。

```
--> fixpix HK500_CDFN2_Domeflat1.fits mdpdb$bpm/nlbpm1_FF64r.fits
```

作成したドームフラットフレームで、A-B スカイ引きしたデータを割ります。結果のファイルは先頭に “fl” を付けることにします。

```
--> !sed 's/\(.*\)/fl\1/' abobj1a.lst > flobj1a.lst
--> imarith @abobj1a.lst / HK500_CDFN2_Domeflat1.fits @flobj1a.lst
```

4.2.4　ゆがみ補正

イメージングデータ用のゆがみ補正データベースを使ってゆがみ補正を行います。geotran のパラメータを以下のように設定します。

```
      input =                Input data
     output =                Output data
   database =                Name of GEOMAP database file
 transforms =                Names of coordinate transforms in
                             database file
(geometry = "geometric")     Transformation type (linear,geometric)
      (xin = INDEF)          X origin of input frame in pixels
      (yin = INDEF)          Y origin of input frame in pixels
   (xshift = INDEF)          X origin shift in pixels
   (yshift = INDEF)          Y origin shift in pixels
     (xout = INDEF)          X origin of output frame in
                             reference units
     (yout = INDEF)          Y origin of output frame in
                             reference units
     (xmag = INDEF)          X scale of input picture in pixels per
                             reference unit
     (ymag = INDEF)          Y scale of input picture in pixels per
                             reference unit
(xrotation = INDEF)          X axis rotation in degrees
(yrotation = INDEF)          Y axis rotation in degrees
     (xmin = INDEF)          Minimum reference x value of output
                             picture
     (xmax = INDEF)          Maximum reference x value of output
                             picture
     (ymin = INDEF)          Minimum reference y value of output
                             picture
```

```
          (ymax = INDEF)        Maximum reference y value of output
                                picture
        (xscale = 1.0)          X scale of output picture in reference
                                units per pixel
        (yscale = 1.0)          Y scale of output picture in reference
                                units per pixel
         (ncols = INDEF)        Number of columns in the output picture
        (nlines = INDEF)        Number of lines in the output picture
       (xsample = 1.0)          Coordinate surface sampling interval
                                in x
       (ysample = 1.0)          Coordinate surface sampling interval
                                in y
  (interpolant = "linear")      Interpolant
     (boundary = "constant")    Boundary extension (nearest,constant,
                                reflect,wrap)
     (constant = 0.0)           Constant boundary extension
(fluxconserve = yes)            Preserve image flux?
      (nxblock = 512)           X dimension of working block size in
                                pixels
      (nyblock = 512)           Y dimension of working block size in
                                pixels
      (verbose = yes)           Print messages about the progress of
                                the task
```

geotran を実行します。結果のファイルは先頭に “gc” を付けることにします。

```
--> !sed 's/\(.*\)/gc\1/' flobj1a.lst > gcobj1a.lst
--> geotran @flobj1a.lst @gcobj1a.lst
 mdpdb$geomap/mcsdistcrr1_feb07new.dbs mcsdistcrr1_feb07new.gmp
```

ここで、geotran のゆがみ補正データベースは、MDPDB に含まれているものを使っています。mdpdb$geomap/mcsdistcrr1_feb07new.dbs、mcsdistcrr1_feb07new.gmp が検出器 1 用、mdpdb$geomap/mcsdistcrr2_feb07new.dbs、mcsdistcrr2_feb07new.gmp が検出器 2 用の database、transform です[3]。

4.3 データの切り出し

次のセクションからの作業は、天体毎に個別に処理をしていきます。そのため、個別の天体のスペクトルを別のファイルに切り出します。

[3] これ以外の時期のデータ用のゆがみ補正データベースが必要な場合は、MCSRED に含まれているものを使用してください。

MCSMDP のタスクである maskplot を使って、MDP ファイルをゆがみ補正済み画像の上にプロットします。

```
—> maskplot CDFN_MASK02.mdp image=gcflabcrMCSA00057147.fits raw+
```

マスクデザインの時と生データでは座標のとり方が異なるので、ここでは大雑把に座標変換して合わせています。そのため、スリットの正確な位置にはプロットされていませんが、どのスペクトルがどの天体かを区別するには役に立つと思います。

この講習では、天体 MODS11-0390 を整約することにします。ファイルリストを作成した後で、imcopy します。切り出す座標は各自確認してください。この時、スリットの両端に関係の無いデータが入らないことはもちろんですが、スリットを短くしすぎないことにも気をつけてください。

```
—> !sed 's/\(.*\)\.fits/\1_MODS11-0390\.fits/' gcobj1a.lst >
 gcMODS11-0390.lst
—> !sed 's/\(.*\)/\1[*,1755:1840]/' gcobj1a.lst > cut.lst
—> imcopy @cut.lst @gcMODS11-0390.lst
```

4.4　個別処理

4.4.1　波長較正

ここでは、スペクトルの X 軸と波長の対応を求めます。近赤外線のデータの場合、観測可能な波長全体に渡って OH の夜光輝線が同時に観測されますのでそれを利用します。ここまで処理してきたデータは、A-B スカイ引きによってスカイの輝線が引き算されてしまっていますので、スカイの輝線を引き算しないスペクトルを作成します。以下のようにします。

```
—> !sed 's/\(.*\)/flsky\1/' crobj1a.lst > flsky1a.lst
—> imarith @crobj1a.lst / HK500_CDFN2_Domeflat1.fits @flsky1a.lst
—> !sed 's/\(.*\)/gc\1/' flsky1a.lst > gcsky1a.lst
—> geotran @flsky1a.lst @gcsky1a.lst
 mdpdb$geomap/mcsdistcrr2_feb07new.dbs mcsdistcrr2_feb07new.gmp
—> !sed 's/\(.*\)\.fits/\1_MODS11-0390\.fits/' gcsky1a.lst
 > gcskyMODS11-0390.lst
—> !sed 's/\(.*\)/\1[*,1755:1840]/' gcsky1a.lst > cut.lst
—> imcopy @cut.lst @gcskyMODS11-0390.lst
```

identify というタスクを使います。identify は、ある行のスペクトルを波長方向に切り出し、その上で検出された輝線の X 座標と、指定した波長のリストの波長を同定します。始めのいくつかを手で指定すると、自動的にフィットして残りの輝線を検出、同定します。パラメータを以下のように設定します。

```
images =                    Images containing features to be
                            identified
```

```
        crval =                 Approximate coordinate (at reference
                                pixel)
        cdelt =                 Approximate dispersion
     (section = "middle line")  Section to apply to two dimensional
                                images
    (database = "database")     Database in which to record feature
                                data
   (coordlist = "mdpdb$ohlist/list_NS_HK500") User coordinate list
       (units = "")             Coordinate units
        (nsum = "20")           Number of lines/columns/bands to
                                sum
                                in 2D images
       (match = -3.0)           Coordinate list matching limit
 (maxfeatures = 50)             Maximum number of features for
                                automatic identification
      (zwidth = 100.0)          Zoom graph width in user units
       (ftype = "emission")     Feature type
      (fwidth = 8.0)            Feature width in pixels
     (cradius = 5.0)            Centering radius in pixels
   (threshold = 0.0)            Feature threshold for centering
      (minsep = 2.0)            Minimum pixel separation
    (function = "chebyshev")    Coordinate function
       (order = 4)              Order of coordinate function
      (sample = "*")            Coordinate sample regions
    (niterate = 10)             Rejection iterations
  (low_reject = 3.0)            Lower rejection sigma
 (high_reject = 3.0)            Upper rejection sigma
        (grow = 0.0)            Rejection growing radius
   (autowrite = no)             Automatically write to database
    (graphics = "stdgraph")     Graphics output device
      (cursor = "")             Graphics cursor input
     (aidpars = "")             Automatic identification algorithm
                                parameters
```

section でどの行を使うか、nsum で何行足し合わせたものを使うかを指定します。coordlist が同定に使用する OH 夜光のリストです。Rousselot et al. (2000) [3] の夜光アトラスから HK500 グリズムの波長較正に適当なものを選んだものが MCSMDP に含まれています（図 4.1)。

以下のように identify を実行します。

```
--> identify gcflskycrMCSA00057147_MODS11-0390.fits
```

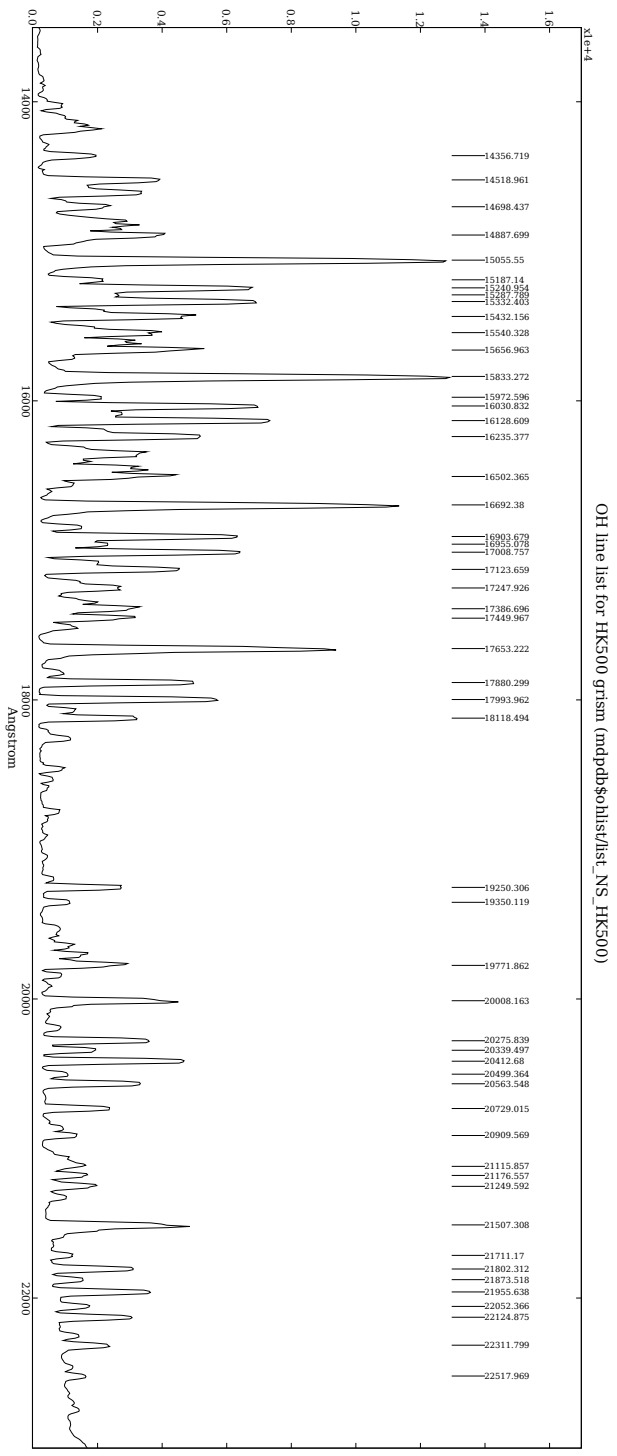

図 4.1: HK500 グリズム用 OH 夜光アトラス

図 4.2 のようなグラフィックウインドウが表示され、対話的に処理を行います。横軸が X 軸のピクセルで、縦軸は、ここでは画像の中心の行から 20 ピクセル幅で足し合わせたカウントになっています。キーボードからコマンドを打って操作しますので、主なコマンドを表 4.1 にまとめておきます。

表 4.1: identify の主なコマンドキー

キー	動作
m	カーソルの近くの輝線を検出し、波長を指定します。
d	カーソルの近くで同定されている輝線を削除します。
c	カーソルの近くで同定されている輝線の座標を表示します。
w	スペースに続けてウインドウコマンドを入力します（下記）。
f	フィットモードに入ります（後述）。
l	リストから自動的に輝線を検出し、同定します。
q	identify を終了します。
?	コマンドのリストをターミナルに表示します。
	ウインドウコマンド
b	カーソルの場所がグラフの下端になります。
t	カーソルの場所がグラフの上端になります。
x	X 軸についてズームします。
y	Y 軸についてズームします。
?	ウインドウコマンドのリストをターミナルに表示します。

まず、“m” で主な輝線を選んで同定していきます。図 4.1 から、目立つ輝線を 5 個くらい選び、それに対応する輝線のそばで “m” を押します[4]。波長を聞かれるので、その輝線の波長を打ち込みます。整数部分まで打ち込むと一番近い数字の輝線をリストから選んでくれるので、小数点以下まで全て打ち込む必要はありません。

5 個ほど選んだ所で “f” を押すと、座標フィッティングのモードに入り、図 4.3 のようになります。横軸が波長、縦軸がフィットしている関数からの残り（residual）です。3 次の Chebychev 多項式でフィットしています[5]。次数やフィットする多項式などはここで対話的に変更することもできます。この座標フィッティングのモードは identify 以外でも iraf で座標の多項式フィットを行う場所では良く使われます。フィッティングモードでの主なコマンドを表 4.2 にまとめておきます。

[4]H バンドにある明るい 4 本は明るすぎてサチることが多いので避けてください。今回は検出器 1 のデータの解析をしていますので、X が大きくなるにつれて波長が小さくなっていることに注意してください。

[5]order=4 となっていますが、iraf の order は次数ではなく項の数を表すので数学で言う次数は order-1 になります。

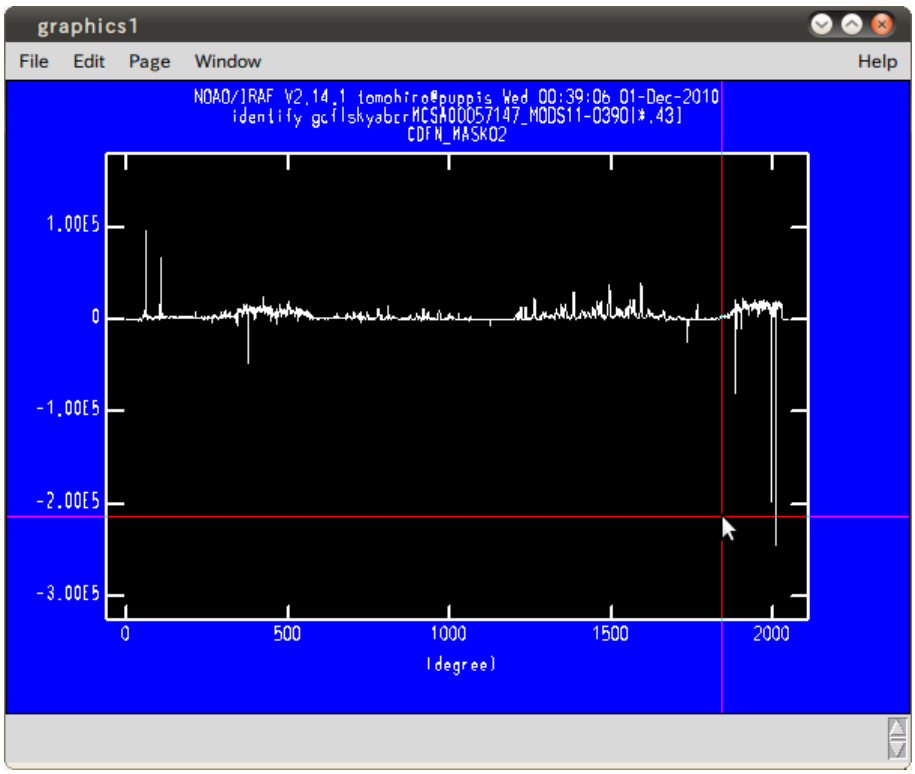

図 4.2: identify のグラフィックウインドウ

表 4.2: identify のフィッティングモードの主なコマンドキー

キー	動作
h,i,j,k,l	グラフの縦横の軸を変えます。
f	データを再フィットしてグラフを再描画します。
d	カーソルの近くの点をフィッティングから除外します。
u	フィッティングから除外した点を復活します。
c	カーソルの近くの点の座標を表示します。
s	フィッティングに使うサンプルを領域指定します。
z	領域指定を解除します。
:order n	フィッティングの次数を n-1 に変更します。
:function 関数名	フィッティングに使う多項式を変更します。
:niterate	sigma rejection を行う回数を指定します。
:low_reject	sigma rejection の下側のしきい値を指定します。
:high_reject	sigma rejection の上側のしきい値を指定します。
q	フィッティングモードを抜けて元の画面に戻ります。
?	フィッティングコマンドのリストをターミナルに表示します。

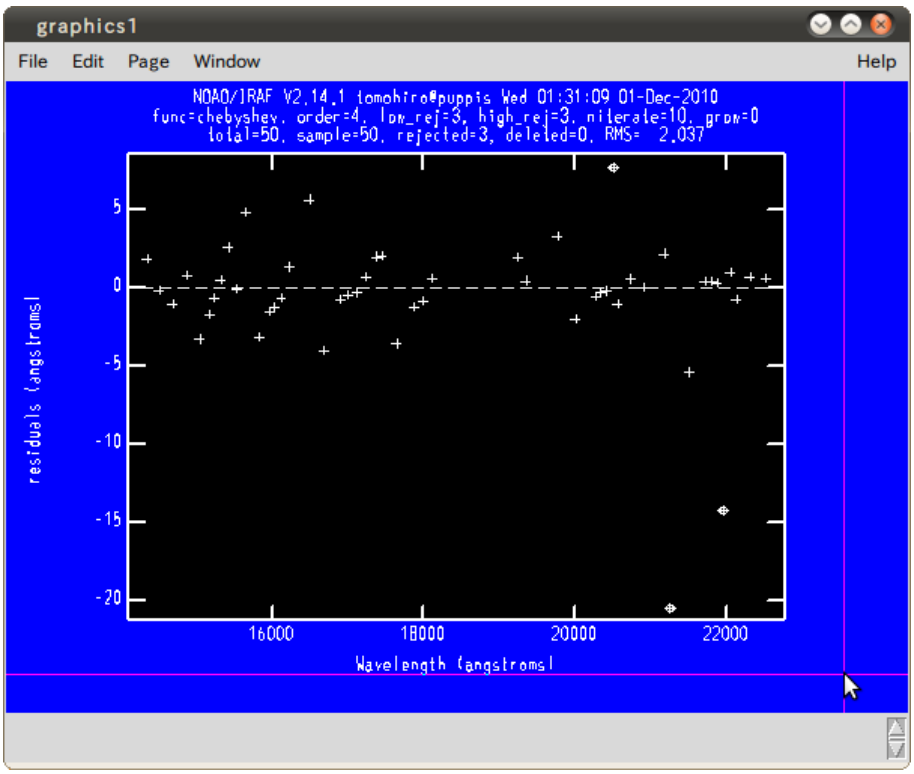

図 4.3: identify のフィットモード

ここで、“q” を押すと最初のように戻りますが、横軸が波長になっていることに気づくでしょう。今、フィットした関数を元に計算された値になっています。さらに “l” を押すと、夜光リストにある残りの輝線を自動的に同定します。

再び “f” を押してフィットモードに入り、良いフィットになっているかどうかを確認します。HK500 グリズムの場合、7.72 [Å/pixel] でしたので、RMS が数Å 程度になっていれば充分です。“q” を押してフィットモードを抜け、もう一度 “q” を押すと identify を終了します。この時、

```
Write feature data to the database (yes)?
```

と聞かれますので、return キーを押します。これによって、database ディレクトリの下のファイル (database/idgcflskycrMCSA00057147_MODS11-0390) に同定された輝線と座標の対応の情報が書き込まれます。

ここまでの作業で、画像の中心の行についてはピクセルと波長の対応が付きましたが、スペクトルの全ての行でその対応が成り立つとは限りません。検出器とスリットの相対的に回転していたり、光学系による像の歪みによってスペクトルが曲がります。

reidentify というタスクを使うと、identify で中心の行について輝線を同定し、フィットしたデータベースを元に、上下の行で自動的に輝線を同定します。reidentify のパラメータを以下のように設定して実行します。

```
    reference =                 Reference image
       images =                 Images to be reidentified
       answer = "yes"           Fit dispersion function
                                interactively?
        crval =                 Approximate coordinate (at reference
                                pixel)
        cdelt =                 Approximate dispersion
(interactive = "no")            Interactive fitting?
    (section = "middle line")   Section to apply to two dimensional
                                images
     (newaps = yes)             Reidentify apertures in images not in
                                reference?
   (override = yes)             Override previous solutions?
      (refit = yes)             Refit coordinate function?

      (trace = yes)             Trace reference image?
       (step = "20")            Step in lines/columns/bands for
                                tracing an image
       (nsum = "20")            Number of lines/columns/bands to sum
      (shift = "0.")            Shift to add to reference features
                                (INDEF to search)
     (search = 0.0)             Search radius
      (nlost = 100)             Maximum number of features which may
                                be lost

    (cradius = 5.0)             Centering radius
  (threshold = 0.0)             Feature threshold for centering
(addfeatures = no)              Add features from a line list?
  (coordlist = "mdpdb$ohlist/list_NS_HK500") User coordinate list
      (match = -3.0)            Coordinate list matching limit
(maxfeatures = 50)              Maximum number of features for
                                automatic identification
     (minsep = 2.0)             Minimum pixel separation

   (database = "database")      Database
   (logfiles = "logfile")       List of log files
   (plotfile = "")              Plot file for residuals
    (verbose = yes)             Verbose output?
   (graphics = "stdgraph")      Graphics output device
     (cursor = "")              Graphics cursor input
```

```
    (aidpars = "")              Automatic identification algorithm
                                parameters
```

```
--> reidentify gcflskycrMCSA00057147_MODS11-0390.fits
gcflskycrMCSA00057147_MODS11-0390.fits
```

これによって、中心から上下に 20 ピクセル置きに足し合わせ、それぞれの行で輝線を同定したものがデータベースに追加されました[6]。つまり、データベースには夜光の場所での X,Y 座標と波長の対応が記述されていることになります。これらの対応する点を使って 2 次元でフィットすることによって、画像自体の X 軸を波長に、Y 軸をスリットの空間方向に対応させることができます。

このフィットを行うタスクが fitcoords です。fitcoords のパラメータを以下のように設定します。

```
      images =                  Images whose coordinates are to be fit
    (fitname = "")              Name for coordinate fit in the
                                database
(interactive = yes)             Fit coordinates interactively?
    (combine = no)              Combine input coordinates for a
                                single fit?
   (database = "database")      Database
  (deletions = "deletions.db") Deletion list file (not used if null)
   (function = "chebyshev")     Type of fitting function
     (xorder = 4)               X order of fitting function
     (yorder = 3)               Y order of fitting function
   (logfiles = "STDOUT,logfile") Log files
   (plotfile = "plotfile")      Plot log file
   (graphics = "stdgraph")      Graphics output device
     (cursor = "")              Graphics cursor input
```

function と xorder について、identify, reidentify で良いフィットになったものを入れています。以下のように実行します。

```
--> fitcoords gcflskycrMCSA00057147_MODS11-0390
```

identify の時と似たようなフィッティングのグラフが現れますが、コマンドが少し違います。fitcoords の主なコマンドを表 4.3 にまとめておきます。fitcoords では、X 軸、Y 軸の座標を変えながらフィットが正しいかどうかを確認し、必要に応じてフィットのパラメータを変更します。

[6]interactive=yes として実行すると具体的に何をやっているのかが良く分かると思います

表 4.3: fitcoords の主なコマンドキー

キー	動作
x*[key]*,y*[key]*	それぞれ、X 軸、Y 軸を *key*（下記）に対応する値に変更します。
r	グラフを再描画します。
f	与えられたパラメータでフィットを実行します。
:order n	フィッティングの次数を n-1 に変更します。
:function 関数名	フィッティングに使う多項式を変更します。
q	fitcoords を終了します。
?	フィッティングコマンドのリストをターミナルに表示します。
	[key] の値
x	X 座標
y	Y 座標
z	同定した輝線の波長
s	フィットした時の波長
r	フィットの残差 (s-z)

例えば、“xxyy” と押してから “r” とすると、横軸が X 座標、縦軸が Y 座標となって同定した輝線の分布を表示します。identify と reidentify がどのように輝線を同定したかがわかります。画像全体に広く分布していることを確認してください。

X 軸のフィットの確認をするには、X 軸を x 座標、Y 軸をフィットの残差にします (“xxyr”)。xorder を変えてフィットしながら、残差の RMS だけでなく全体に大域的なパターンが無いかを確認してください。例えば 2 次関数のようなパターンがある場合、次数を 2 つ上げるとそのパターンが消えます。大域的なパターンが出ない、最も低い次数になるように指定します。同様に、X 軸を y 座標、Y 軸をフィットの残差にする（“xyyr”）と、Y 軸のフィットの確認ができます。確認が終わったら、q を押して fitcoords を終了し、結果をデータベースに書き出します。

ここまでの作業では、1 枚目のフレーム (MCSA00057147) から検出した夜光輝線を使って変換用のフィット関数を求めました。MOIRCS では装置のたわみによって観測中に最大で 3 ピクセル程度、マスクと検出器の相対位置が移動することが知られています。従って、厳密には全てのフレームで夜光輝線の同定をして場所を合わせる必要があります[7]。フレーム数もそれほど多くないので、ここでは 1 枚目のフレームで求めたフィットを元に、全てのフレームのゆがみ補正をします。

fitcoords で求めたデータベースを元に画像の変換を行うタスクが transform です。transform のパラメータを以下のように設定します。

```
      input =                 Input images
     output =                 Output images
   fitnames =                 Names of coordinate fits in the
                              database
  (minput = "")               Input masks
```

[7]MCSMDP は自動的にそれを行うようになっています。

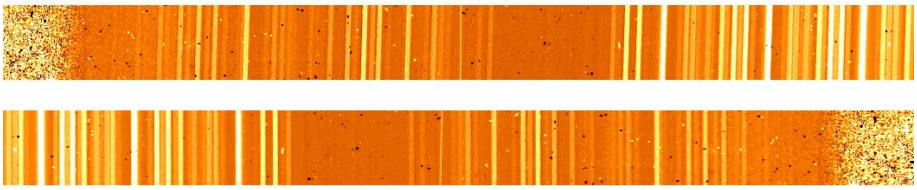

図 4.4: ゆがみ補正前後の 2 次元スペクトル（上:補正前、下：補正後）

```
    (moutput = "")              Output masks
   (database = "database")      Identify database
(interptype = "linear")         Interpolation type
         (x1 = INDEF)           Output starting x coordinate
         (x2 = INDEF)           Output ending x coordinate
         (dx = INDEF)           Output X pixel interval
         (nx = INDEF)           Number of output x pixels
       (xlog = no)              Logarithmic x coordinate?
         (y1 = INDEF)           Output starting y coordinate
         (y2 = INDEF)           Output ending y coordinate
         (dy = INDEF)           Output Y pixel interval
         (ny = INDEF)           Number of output y pixels
       (ylog = no)              Logarithmic y coordinate?
       (flux = yes)             Conserve flux per pixel?
      (blank = INDEF)           Value for out of range pixels
(logfiles = "STDOUT,logfile") List of log files
```

以下のように transform を実行します。結果ファイルの先頭には “tr” が付くことにします。

```
--> !sed 's/\(.*\)/tr\1/' gcMODS11-0390.lst > trMODS11-0390.lst
--> transform @gcMODS11-0390.lst @trMODS11-0390.lst
 gcflskycrMCSA00057147_MODS11-0390
```

transform 前後の画像を ds9 に表示して比べてみましょう（図 4.4）。補正前は夜光輝線が斜めになっていますが、補正後は Y 軸に沿ってまっすぐになっていることがわかります。分散の方向も逆になっていることに気がつくと思います。また、transform した後の FITS ファイルにはヘッダに波長較正の情報が追加されるので、カーソルを当てた場所の波長が ds9 の information panel に表示されるようになります。

```
--> mdpdisplay gcflabcrMCSA00057147_MODS11-0390.fits frame=1
--> mdpdisplay trgcflabcrMCSA00057147_MODS11-0390.fits frame=2
```

4.4.2　残りスカイ引き

切り出しの前に A-B ペアでスカイを引いたにもかかわらず、まだ夜光輝線が残っているように見えます。これは、A-B ペアの観測をしている間にスカイのカウントが変化してしまったことが原因です。ここでは、この引き残ったスカイを空間方向に多項式でフィットして引きます。前の節で、transform を使って Y 軸が空間方向に一致するように補正したのには、このスカイ引きをやりやすくするという目的もあります。これによって、Y 軸方向にフィットするだけでスカイを引くことができるからです。

background というタスクを使います。以下のように background のパラメータを設定し、実行します。結果のファイルには先頭に “bg” が付きます。

```
        input =                 Input images to be background
                                subtracted
       output =                 Output background subtracted images
        (axis = 2)              Axis along which background is fit
                                and subtracted
 (interactive = yes)            Set fitting parameters interactively?
      (sample = "*")            Sample of points to use in fit
    (naverage = 1)              Number of points in sample averaging
    (function = "chebyshev")    Fitting function
       (order = 3)              Order of fitting function
  (low_reject = 3.0)            Low rejection in sigma of fit
 (high_reject = 3.0)            High rejection in sigma of fit
    (niterate = 3)              Number of rejection iterations
        (grow = 0.0)            Rejection growing radius
    (graphics = "stdgraph")     Graphics output device
      (cursor = "")             Graphics cursor input
```

```
--> !sed 's/\(.*\)/bg\1/' trMODS11-0390.lst > bgMODS11-0390.lst
--> background @trMODS11-0390.lst @bgMODS11-0390.lst
```

実行するとグラフ窓が現れて Fit column を聞かれます。ここでは、フィットの条件を決めるのに確認するための X 座標を入力します。ds9 に表示されたスペクトルを見ながら、スペクトルの中心付近で、明るい夜光輝線が残っている列を指定します。

数字を入力すると、指定した列に沿って 1 次元に切り出したカウントと、タスクのパラメータに指定した条件でフィットした関数がプロットされます。このプロットを見ながらフィッティングの条件を指定します。background のグラフ窓で使用するコマンドは identify のフィッティングモードとほぼ同じです。

引け残りのスカイは、傾きに加えて緩やかな弯曲が見える程度が多いので、フィットする関数は 2 次 (order=3) のフィットで良いくらいだと思います。明るい連続光が見えている場合（標準星など）や、スペクトルの切り出しの時に端の方に関係ないデータが入ってしまった場合等は、フィットがそれらに引っ張られてしまいますので、“s” でフィットに使う領域を指定するのが良いです。

フィットに使う領域指定も含めて、ここで指定した条件は全ての列で同じ条件が使われます。1つの列で指定した後 “q” で抜けますが、再び Fit column を聞かれます。他の column でも問題ないことを確認してください。

終わったら、結果の画像を ds9 に表示してきれいにスカイが引けていることを確認します。

4.5 足し合わせ

スペクトルを足し合わせます。今、処理している 4 枚のデータは、それぞれ A-B のペアでスカイ引きをしたものでした。つまり、A 点のデータだけでなく、B 点の分のデータはネガティブなシグナルとして写っているはずです。そこで、プラスマイナスをひっくり返した画像を作りディザした分だけずらすと、A 点のデータに加えて B 点のデータも足すことができ、8 枚分のシグナルを足すことができます。

まず、ディザ幅分ずらして B 点のシグナルを A 点と同じ場所に移動します。ディザ幅はヘッダに書かれているので、以下のように確認します。

```
--> hselect @bgMODS11-0390.lst "$I,K_DITWID" yes
bgtrgcflabcrMCSA00057147_MODS11-0390.fits        3.000
bgtrgcflabcrMCSA00057151_MODS11-0390.fits        3.000
bgtrgcflabcrMCSA00057159_MODS11-0390.fits        3.000
bgtrgcflabcrMCSA00057163_MODS11-0390.fits        3.000
```

すべて 3 秒のディザを振っていることがわかりました。MOIRCS のピクセルスケールは 0.117 arcsec/pixel なので、3.0/0.117 ~ 26pixel ずらします。整数にしたのはサブピクセルの移動があると余計な内挿が入るためです。imshift を使います。

```
--> !sed 's/\(.*\)/sh\1/' bgMODS11-0390.lst > shMODS11-0390.lst
--> imshift @bgMODS11-0390.lst @shMODS11-0390.lst 0 26
```

次に、シフトした画像の符号をひっくり返します。imarith を使います。

```
--> !sed 's/\(.*\)/ng\1/' shMODS11-0390.lst > ngMODS11-0390.lst
--> imarith @shMODS11-0390.lst * -1 @ngMODS11-0390.lst
```

最後に、imcombine を使って画像を足し合わせます。今回は画像の枚数がそれほど多くないのと、潰しきれていない宇宙線イベントがかなりの数ありますので、足し合わせは median を取ることにしました。以下のように imcombine のパラメータを設定し、実行します。

```
        input = ""              List of images to combine
       output = ""              List of output images
  (headers = "")                List of header files (optional)
  (bpmasks = "")                List of bad pixel masks (optional)
 (rejmasks = "")                List of rejection masks (optional)
(nrejmasks = "")                List of number rejected masks
```

```
      (optional)
  (expmasks = "")         List of exposure masks (optional)
    (sigmas = "")         List of sigma images (optional)
     (imcmb = "$I")       Keyword for IMCMB keywords
   (logfile = "STDOUT")   Log file

   (combine = "median")   Type of combine operation
    (reject = "sigclip")  Type of rejection
   (project = no)         Project highest dimension of input
                          images?
   (outtype = "real")     Output image pixel datatype
  (outlimits = "")        Output limits (x1 x2 y1 y2 ...)
    (offsets = "none")    Input image offsets
  (masktype = "none")     Mask type
  (maskvalue = "0")       Mask value
      (blank = 0.0)       Value if there are no pixels

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

(lthreshold = INDEF)      Lower threshold
(hthreshold = 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 = 3.0)       Lower sigma clipping factor
     (hsigma = 3.0)       Upper sigma clipping factor
    (rdnoise = "0.")      ccdclip: CCD readout noise (electrons)
       (gain = "1.")      ccdclip: CCD gain (electrons/DN)
     (snoise = "0.")      ccdclip: Sensitivity noise (fraction)
   (sigscale = 0.1)       Tolerance for sigma clipping scaling
                          corrections
      (pclip = -0.5)      pclip: Percentile clipping parameter
       (grow = 0.0)       Radius (pixels) for neighbor rejection
       (mode = "al")
```

```
--> imcombine @bgMODS11-0390.lst ,@ngMODS11-0390.lst HK500_MODS11-0390
```

後は、各天体について同じように天体の切り出し、波長較正とゆがみ補正、足し合わせを行います。

4.6 フラックス較正

スペクトルデータの値は、1ピクセル当たり、1露出当たりのカウント（ADU）になっています。天体からのフラックスを測定するためには、これを波長当たりのフラックス (F_λ [erg sec^{-1} cm^{-2} Å^{-1}]) にします。このため、あらかじめ明るさが分かっている標準星をターゲット天体を観測したのと近い時間で、ターゲット天体とほぼ同じ高度にあるものを観測します。

大気の吸収や望遠鏡・装置等の効率などによって、宇宙からやってきた光子は全てが検出器に届くわけではありません。これらは、波長に依存すると考えられますので、特に分光観測の場合は波長の関数としてカウントとフラックスの関係を求める必要があります。観測されるカウントを $N_{\rm obs}(\lambda)$、天体の本来のフラックスを $F_{\lambda,{\rm int}}$、大気の吸収や望遠鏡・装置等の効率を $R(\lambda)$ と置くと、以下のような関係であると考えます。

$$N_{\rm obs}(\lambda) = R(\lambda) \times F_{\lambda,{\rm int}} \tag{4.1}$$

可視の場合、天体の本来のフラックスについて、多くの分光標準星のライブラリができていますが、近赤外線の場合はそれがありません。そこで、スペクトル型がわかっている星を観測し、そのスペクトル型の理論的なモデルを使用します。水素の吸収線を除けば比較的のっぺりしたスペクトルを持つ A 型星がしばしば使われます。モデルスペクトルは、今回は Castelli (2004) [1] を使って作成したものを用意しました[8]。

まず、標準星のデータをオブジェクトの時と同じように整約して 2 次元スペクトルを作ります。前の節までを参考にして作業してください。オブジェクトのデータと異なり強い連続光が受かっているので、スカイ引きの時に天体の光の部分を避けることなどに気をつけてください。作成した 2 次元スペクトルのファイル名を HK500_M53735.fits とします。

2 次元スペクトルから連続光の部分を切り出すためには、apall というタスクを使います。apall は、主に 2 つの処理を行います。1 つ目が連続光を切り出すアパーチャと背景部分の決定、2 つ目が連続光のトレースと切り出しです。それぞれのパラメータは対話的に決めますので、デフォルトのまま、以下のとおり実行してください。いろいろ聞かれますが、基本的に 'yes' で答えれば良いです。

```
--> apall HK500_M53735.fits
Recenter apertures for HK500_M53735? ('yes'):
Resize apertures for HK500_M53735? ('yes'):
Edit apertures for HK500_M53735? ('yes'):
```

[8] 水素の吸収線を気にしなければ黒体輻射でも可能です。モデルの吸収線の幅は、観測の波長分解能に応じてなまらせる必要がありますが、今回の観測用に作成したものを用意しています。

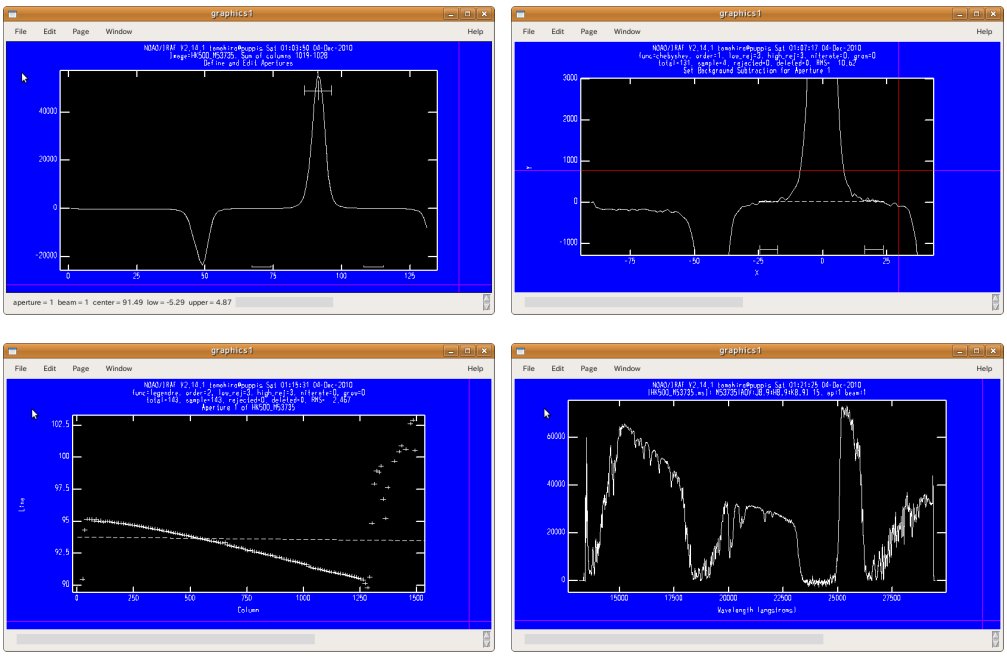

図 4.5: apall のグラフ画面。(左上: アパーチャーとバックグラウンドを決めるグラフの初期画面。右上: バックグラウンドを決める画面。左下: スペクトルをトレースする多項式を決める画面。右下: 最終的に得られたスペクトル。)

まず、アパーチャーとバックグラウンドを決めるためのグラフが表示されます（図 4.5 左上）。アパーチャーは自動的に検出され、決められますので、“b” を押してバックグラウンドを設定します（図 4.5 右上）。この画面では、バックグラウンドを設定する場所を “s” で指定します。アパーチャの両側の適当な場所を指定します。“q” で最初のグラフに戻り、もう一度 “q” でアパーチャを決めるグラフを終了します。

```
Trace apertures for HK500_M53735?
Fit traced positions for HK500_M53735 interactively?
Fit curve to aperture 1 of HK500_M53735 interactively
```

次に、スペクトルをトレースするグラフが表示されます（図 4.5 左下）。検出されたアパーチャの頂点が多項式でフィットされます。スペクトルの切り出しはこの多項式に沿って行われますので、“s” でフィットに使う範囲を、“:order” でフィットの次数等を調整して、残差が小さくなるようにします。

```
Write apertures for HK500_M53735 to database?
Extract aperture spectra for HK500_M53735?
Review extracted spectra from HK500_M53735?
```

最後に、切り出したスペクトルが表示されます (図 4.5 右下)。結果のファイルは、HK500_M53735.ms.fits のような名前になっているはずです。これが、上の式の $N(\lambda)$ になります。

次に、これをモデルスペクトルで割って $R(\lambda)$ を求めます。MCSMDP に含まれるタスク、rescurve を使用します。以下のようにパラメータを指定します。

```
    stddata = "" one-D standard star data
     stdmag =    Vega magnitude of the standard at magfilter
     output = "" output response curve
(magfilter = "mdpdb$filter/2mass_K.fits") transmission curve of filter for stdmag
  (moddata = "mdpdb$standard/A0V_g50.spc") standard star model spectrum
(vegamodel = "mdpdb$filter/alpha_lyr_stis_003.fits") Vega model spectrum
```

モデルスペクトルのデータ (A0V_g50.spc) の他に、K バンドでの Vega 等級と等級を決めるためのフィルター曲線、Vega 等級からフラックスに直すための Vega モデルスペクトルを指定します。全て MDPDB の中に含まれています。

今回は、HIPPARCOS 星表の A0 型星、M53735 を観測しました。K バンドでの等級は 2MASS のカタログから 8.856 とわかります。そこで、以下のように実行します。

```
--> rescurve HK500_M53735.ms.fits 8.856 resc_CDFN2_HK500.fits
```

resc_CDFN2_HK500.fits が、$R(\lambda)$ になります[9]。

最後に、オブジェクトの 2 次元スペクトルを $R(\lambda)$ で割って、フラックス較正した 2 次元スペクトルを得ます。MCSMDP のタスク、mdpfcalib を使います。このタスクは、同時に露出時間で割って 1 秒当たりのエネルギー (=フラックス) に直します。

```
--> mdpfcalib HK500_MODS11-0390.fits resc_CDFN2_HK500.fits
HK500_MODS11-0390fl.fits
```

フラックス較正済みデータ、HK500_MODS11-0390fl.fits の単位は、[erg sec^{-1} cm^{-2} Å^{-1}] になっています。

[9]バイナリテーブル FITS になっているので ds9 で直接見ることはできません。

第5章　データ解析実習

5.1　赤方偏移を求める

この観測では、多色の測光データを用いて、$z \sim 2$ の星形成銀河が持つ星からの連続光の特徴をとらえるカラーによって観測する天体を選んでいます (Yoshikawa et al. 2010 [5])。この観測の目的は、これらの高赤方偏移の銀河の輝線を調べることです。まずは、足し合わされた 2 次元スペクトルから輝線を検出するとともに、そこから天体の赤方偏移を求めます。

まず、2 次元スペクトルを ds9 に表示して、輝線を探してください。このフレームは A-B の引き算をしているので、本物の輝線であればポジティブで受かっている輝線の上下にネガティブに 2 つの輝線が見えるはずです。輝線と思われるものが見つかったら、その波長と空間方向の座標（Y 座標）をメモしてください。

—> mdpdisplay HK500_MODS11−0390fl.fits frame=1

次に、輝線の重心の波長を求めます。splot というタスクを使うと、2 次元スペクトルを 1 次元化してプロットし、その上でスペクトルの様々な処理を行うことができます。まずは、以下のように splot を実行します。

—> splot HK500_MODS11−0390fl.fits （輝線を検出した空間方向の座標）

splot の主なコマンドを表 5.1 にまとめます。最初の画面では、引数で指定した座標について、1 ピクセル幅で切り出したスペクトルを、横軸を波長として表示しています。そこで、まずは “:nsum” で空間方向に足し合わせます。ds9 に表示した 2 次元スペクトルを見ながら適当な幅を決めますが、5 か 7 ピクセル程度が良いでしょう。

次に、ウインドウコマンド (“w”) や、“a” を使って先ほど輝線を検出した波長のところを拡大します。目的の輝線が表示されたら、“k” を使って測定します。輝線の左側と右側にカーソルを持っていって “k” を押すと、それぞれの場所をゼロ（連続光または背景）として、その間にある輝線を Gaussian でフィットします。フィット結果はグラフのウィンドウ下部に、中心波長、フラックス、等価幅、FWHM の順で表示されます。

HK500 グリズムでとらえられる波長は 1.3 μm から 2.5 μm だったので、この銀河が実際に $z \sim 2$ の銀河だとすると、静止系の波長で 6000Å 付近の可視光の輝線を検出することができるはずです。表 5.2 に銀河、AGN で検出される可視域の主な波長をリストしておきます[1]。少なくとも 2 つの輝線が受かっているはずですので、この表の輝線と実際に受かった輝線の波長を比べながら、この天体の赤方偏移を推定し、それぞれ何の輝線かを同定してください。

[1] それぞれの科学的な意味に付いてはこのテキストの範囲を超えるので省略します。

表 5.1: splot の主なコマンドキー

キー	動作
a	波長に対して拡大する。拡大表示したい波長の左端と右端にカーソルを移動し、それぞれで “a” を押す。
w	スペースに続けてウインドウコマンドを入力します（下記）。
k	輝線の Gaussian フィット。フィットした輝線の左側と右側にカーソルを移動し、それぞれで “k” を押す。
s	スペクトルにスムージングをかける。（スムージングのカーネルサイズを聞かれる）
q	fitcoords を終了します。
?	フィッティングコマンドのリストをターミナルに表示します。
:nsum n	空間方向に n ピクセル足し合わせます。

表 5.2: 主な可視域の輝線

輝線	波長 [Å]	備考
[SII]	6716, 6731	強度比 1:1
Hα	6563	水素のバルマー系列
[NII]	6548, 6583	Hα の両脇、強度比 1:2
[OIII]	4959, 5007	強度比 1:3
Hβ	4861	水素のバルマー系列

5.2 赤方偏移とフラックスから光度を求める

splot で輝線の波長、フラックスを求めましたので、Hα 輝線の光度（単位時間当たりに放出されるエネルギー）を求めてみましょう。フラックス (F) と光度 (L) の関係は、以下のように与えられます。

$$L = F \times 4\pi d_L^2(z) \tag{5.1}$$

$d_L(z)$ は光度距離で、宇宙論を仮定すると赤方偏移の関数として求まります。光度距離の詳細は、宇宙論の教科書などを参照してください。MCSMDP に、光度距離の計算をするタスクを作ってありますので今回はそれを利用します。赤方偏移が 2 の場合、以下のように実行し、1 つ目が光度距離です。

```
--> cosmology 2.0
Luminosity Distance(m): 4.800e+26
Angular Diameter Distance(m): 5.333e+25
Look-Back Time(yr): 1.024e+10
```

参考・輝線の光度について

輝線の光度について非常に単純な部分だけをまとめましたが、これだけでは不十分です。例えば、Hα の光度から銀河の星形成率を求める場合、以下のような点に気をつける必要があります。

アパーチャ

今、空間方向に適当なアパーチャで潰して 1 次元スペクトルにしましたが、これが銀河のどこを見ているのかを考慮に入れる必要があります。また、スリットの幅で制限されてしまっている分があるかもしれません。

銀河系での星間減光

我々の銀河系内でどの程度星間塵による星間減光を受けているかを調べます。遠赤外線放射などから、銀河系内のダスト量の分布は知られています。NED の Extinction Calculator [2] で調べることができます。深探査領域の場合、元々星間減光が少ない場所が選ばれている場合が多く、近赤外線での吸収量も小さいので、気にならない場合が多いと思われます。今回の領域では $E(B-V) = 0.011$ 等程度です。

目標の銀河での星間減光

観測されているフラックスは目標の銀河の中の星間塵によって吸収されていることが考えられます。特に、高赤方偏移の銀河が行う激しい星形成によって、近傍よりも多くの塵が生成されていると考えられるので、この効果を調べることは重要です。

[2] http://nedwww.ipac.caltech.edu/forms/calculator.html

参考文献

[1] Castelli, F., Kurucz, R. L. 2004, arXiv:astro-ph/0405087

[2] McLean, 1997, Electronic Imaging in Astronomy; Detectors and Instrumentation (John Wiley & Sons Ltd., 1997)

[3] Rousselot et al. 2000, A&A, 354, 1134

[4] Suzuki, R., et al. 2008, PASJ, 60, 1347

[5] Yoshikawa, T., et al. 2010, ApJ, 718, 112

付 録A　MCSMDPによる自動リダクション

A.1 全体の流れと MCSMDP の特徴

ここでは、MCSMDP のスクリプトを使って半自動で分光データを整約する方法について説明します。基本的な流れは図 3.1 と同じですが、共通処理、個別処理の部分の処理にそれぞれ mdpproc, mdpcombine というタスクを使用します。

MCSMDP は、本テキスト中で解説した処理に加え、作業効率を上げる、輝線検出の S/N を上げる、エラーを評価するなどの目的で以下のような工夫をしています。

宇宙線・バッドピクセル処理 本文中「参考」に示したように、A-B ペアを利用した宇宙線検出、検出象限境界のピクセルの補完処理を行う

スペクトルの切り出し MDP ファイルの座標を利用してスペクトルの自動切り出しを行う。

波長較正:各フレームでの OH 夜光同定 ある天体のスペクトルで 1 つ目の天体の OH 夜光を同定すると、それをリファレンスにしてその他のフレームでもそれぞれで同定を行う。観測中に波長方向にマスク-検出器の相対位置がずれた分に関して自動的に補正する。

波長較正:自動 OH 夜光同定 1 つ目の天体に対して手動で OH 夜光の同定を行った後は、それをリファレンスにして他の天体では自動で同定を行う。

スカイフレームの作成 共通処理、個別処理の両方でスカイ引きをしないフレームを作成し、multi-extention FITS (MEF) ファイルとして 2 つ目の extention にスカイフレームを付加する。

エラーの推定 フラックス較正の時にスカイフレームのカウントを元にスカイの Poisson ノイズを推定し、3 つ目の extention にノイズフレームを付加する。

SPECFIT による輝線フィット STSDAS に含まれる specfit というタスクを利用して [NII]-Hα のように HK500 グリズムでは分離が難しい輝線を複数の Gaussian で同時にフィットする。フラックス等のエラーの推定にスカイの Poisson ノイズを利用する。

エラーの推定について補足しておきます。MOIRCS で他天体分光に使用するスリットは、なるべく多くの天体をマスクに載せるためにエラーを推定するのに充分な長さを取ることができません。そこで、スカイ引きをしないフレームを作成して、その Poisson エラーを誤差の推定値として採用します。

ノイズフレームでは以下のような計算式でエラーを推定します。

$$\sigma_i = \frac{\sqrt{g \times x_i}}{g} \times \frac{1}{r} = \frac{1}{r}\sqrt{\frac{x_i}{g}}, \tag{A.1}$$

ここで、x_i はスカイフレームのカウント、g は検出器のゲイン（$\mathrm{e^-ADU^{-1}}$）、r は response curve（$\mathrm{ADU\,erg^{-1}\,sec\,cm^2\,\AA}$）です。

さらに、1 次元スペクトルに潰したとき、以下のような計算式でエラーを推定します。

$$\sigma_{\mathrm{obj}} = \sqrt{\frac{2\Sigma_{\mathrm{obj}}\sigma_i^2}{t_{\mathrm{exp}}}}, \tag{A.2}$$

t_{exp} は露出時間、係数 $\sqrt{2}$ は、A-B スカイ引き分を考慮しているためです。

A.2 実際の処理

A.2.1 フラットフレーム

mkdflat というタスクを使用します。各パラメータの意味は以下のとおり。

infile 入力ファイルリスト

outfile 出力ファイル名

bpm バッドピクセルマスク (MDPDB にあるものを使う)

offfile ドームライトがオフの時のフレーム。無いときは省略可能。

normilize フラットの値が大きすぎないように規格化しておく値。全体の imstat を取ったりするとスリットの密度によって値が代わってしまうので、適当に指定します。

以下のように実行します。

```
--> mkdflat @flat1.lst HK500_CDFN2_Domeflat1.fits
 mdpdb$bpm/nlbpm1_FF64r.fits
```

A.2.2 テンプレートファイル作り

まず、mktemplate というスクリプトを使ってリダクションのテンプレートファイルを作ります。基本的には、ここで作ったテンプレートファイルの内容を pyraf 環境の中で実行していきます。

まず、mktemplate の設定ファイルを編集します。MCSMDP のインストールディレクトリにある mktemplate.conf をカレントディレクトリにコピーし、それを編集します。そのあと、その設定ファイルを引数として mktemplate を実行すると、標準出力にテンプレートが出力されるので適当なファイルに保存してください。

```
--> copy MCSMDP$doc/mktemplate.conf .
--> !vi mktemplate.conf
--> mktemplate mktemplate.conf > log.py
```

mktemplate.conf ファイルの中身は以下の通りです。bash のスクリプトで変数に値を入れる場合と同じ文法です。“=” の前後にはスペースを入れないように、“$” を入れるときはエスケープシーケンスを使うようにしてください。

```
## parameters
coordlist="mdpdb\$ohlist/list_NS_HK500"
mdpfile="CDFN_MASK02.mdp"
prefix="HK500"
# calibration data for chip1
bpm1="mdpdb\$bpm/nlbpm1_FF64r.fits"
```

```
flat1="HK500_CDFN2_Domeflat1.fits"
gdb1="mdpdb\$geomap/mcsdistcrr1_feb07new.dbs"
gmp1="mcsdistcrr1_feb07new.gmp"
# calibration data for chip2
bpm2="mdpdb\$bpm/nlbpm2_FF64r.fits"
flat2="HK500_CDFN2_Domeflat2.fits"
gdb2="mdpdb\$geomap/mcsdistcrr2_feb07new.dbs"
gmp2=" mcsdistcrr2_feb07new.gmp"
```

各変数の意味は以下の通りです。

coordlist 波長較正に使うリスト

mdpfile マスクデザインファイル

prefix 最終 2 次元スペクトルファイルの頭に付く文字列。prefix + "_" + MDP ファイル上の名前 + "fits"になる

bpm[12] バッドピクセルマスク名

flat[12] フラットファイル名

A.2.3 共通処理

mktemplate.sh が生成したファイルのとおり実行します。

以下の例のように、各チャンネル、ディザ点ごとにファイルリストを用意します。AB が対応するようにリストの中の順番に注意してください。観測時の都合で、A だけ (B だけ) になっている場合は、重複しても構わないので時間が近い B(A) 点フレームを並べてください。

```
hselect @files.list $I "@'DET-ID' = 1 & K_DITCNT = 1" > files1a.list
hselect @files.list $I "@'DET-ID' = 1 & K_DITCNT = 2" > files1b.list
hselect @files.list $I "@'DET-ID' = 2 & K_DITCNT = 1" > files2a.list
hselect @files.list $I "@'DET-ID' = 2 & K_DITCNT = 2" > files2b.list
```

mdpproc を実行します。

```
unlearn mdpproc
iraf.mdpproc.bpmask="mdpdb$bpm/nlbpm1_FF64r.fits"
iraf.mdpproc.gdb="mdpdb$geomap/mcsdistcrr1_feb07new.dbs"
iraf.mdpproc.gmp="mcsdistcrr1_feb07new.gmp"
iraf.mdpproc.flat="HK500_CDFN2_Domeflat1.fits"
mdpproc @obj1a.lst @obj1b.lst
```

結果ファイルは Multi Extention FITS (MEF) ファイルになっていて、0 番が ab スカイ引き済みのデータ、1 番がスカイを引いていないデータ (ノイズ見積り、波長較正用) になっています。

A.2.4 データの切り出し

mdptran を使用して MDP ファイルの座標を FITS ファイルの座標に合わせます。以下のように実行します。

```
—> mdptran CDFN_MASK02.mdp CDFN_MASK02tr1.mdp
 gcflabcrMCSA00057147.fits
```

1 つ目の引数がオリジナルの MDP ファイル、2 つ目が変換後の MDP ファイルのファイル名、3 つ目が基準にする画像の FITS ファイルです。

実行すると、maskplot と同様に ds9 に FITS ファイルが表示されてその上にスリットの大雑把な場所がプロットされます。ターミナルに、

```
select shape then return or 'q' to quit:
```

と、表示されますので、適当なスリットをクリックして選択状態にします。選択したら、ターミナルの上で return キーを押します。すると、ターミナルに、

```
click the object position:
```

と、表示されますので、ds9 の上で先ほどのスリットに対応するスペクトルから、明るい夜光の真ん中をクリックします（HK500 グリズムの場合、16692Å の夜光が一番良いです)。すると、最初と同じようにスリットの選択に戻るので、同様にスリットと夜光の対応をクリックで指定していってください。この時、夜光は必ず同じ波長の物を選ぶようにしてください。

スリットは全て選ぶ必要は無く、最低でも 3 個、4 ▪ 5 個くらい選べば良いです。また、画像の中から偏らずにまんべんなく選ぶようにしてください。適当な数を選んだら、スリットの選択待ちの時に ‘q’+return を押します。すると、MDP ファイルの座標が調整され、ds9 の上にスリットがプロットしなおされます。結果に問題なければ、

```
type q to write mdp database to file:
```

に、‘q’+return を押してください。MDP ファイルに書き出されます。この時、q 以外を押すと結果が破棄されて最初に戻るので注意してください。

この座標を合わせた MDP ファイルを使って cutspec というタスクを使うと、以下のようにオブジェクト名を指定するだけでスペクトルを切り出すことができます。また、次の個別処理で使う mdpproc は、同様にオブジェクト名から自動的にスペクトルを切り出します。

```
—> cutspec @gcobj1a.lst CDFN_MASK02tr1.mdp name=ALE03_205
```

A.2.5 個別処理

mdpcombine を使ってどれか一つのスペクトルの波長較正を手動で行います。mdptran によって ds9 にプロットされたオブジェクト名を見ながら、真ん中辺りのスリットを選びます。4 つ目の引数に手動で波長較正するスリットのオブジェクト名を指定します。

```
--> unlearn mdpcombine
--> iraf.mdpcombine.coordlist="mdpdb$ohlist/list_NS_HK500"
--> iraf.mdpcombine.ymin=42
--> iraf.mdpcombine.ymax=2003
--> mdpcombine @gcflab1.list CDFN_MASK02tr1.mdp HK500_MODS11-0390
 MODS11-0390 id_mode=manual
```

1つ目を “id_mode=manual” で実行した後は、これをリファレンスにして自動的に OH 夜光を同定します。mktemplate の出力には以下のような sed コマンドが記述されています。これは、MDP ファイルの記述から、mdpcombine で実行すべきコマンドを生成するものです。

```
!awk '{print $8}' CDFN_MASK02tr1.mdp | sed 's/\(.*\)/mdpcombine\
 \@gcflab1\.list\ 'CDFN_MASK02'tr1\.mdp\ 'HK500'_\1 \1/'
```

以下のように mdpcombine のパラメータを設定した後で、各オブジェクトに対する mdpcombine の行を順番に実行していきます。

```
--> iraf.mdpcombine.id_mode='auto'
--> iraf.mdpcombine.refname='MODS11-0390'
--> iraf.mdpcombine.bg_inter=no
--> mdpcombine @gcflab1.list CDFN_MASK02tr1.mdp HK500_MODS11-0094
 MODS11-0094
```

A.2.6 標準星解析、フラックス較正

共通処理は通常の MOS データと同じように mdpproc を使って処理します。

mdpcombine もほとんど同じように使いますが、MDP ファイルは指定せずに直接 ymin, ymax でスリット上下の座標を設定してスペクトルを切り出します。

```
--> unlearn mdpproc
--> iraf.mdpproc.bpmask="mdpdb$bpm/nlbpm2_FF64r.fits"
--> iraf.mdpproc.gdb="mdpdb$geomap/mcsdistcrr2_feb07new.dbs"
--> iraf.mdpproc.gmp="mcsdistcrr2_feb07new.gmp"
--> iraf.mdpproc.flat="HK500_CDFN2_Domeflat2.fits"
--> mdpproc MCSA00057114.fits MCSA00057116.fits
--> unlearn mdpcombine
--> iraf.mdpcombine.coordlist="mdpdb$ohlist/list_NS_HK500"
--> mdpcombine gcflabMCSA00057114.fits INDEF HK500_M53735 M53735
 id_mode=manual coordlist="mdpdb$ohlist/list_NS_HK500"
 calframe="INDEF" ymin=900 ymax=1030
```

標準星のデータは光量の少ないオブジェクトのデータに比べて生データでもスペクトルが明るいので、気を付ける点がいくつかあります。

- idenfity で波長較正をするときに、夜光を同定する（検出器上の）行にスペクトルが載っていると、夜光を見つけられません。id_section パラメータで、”line #”として夜光の同定に使う行を手で指定してください。

- background で引け残り夜光を引く時に、背景をフィットする範囲を手で指定してください。”s”で指定します。

この後、apall で 1 次元化し、rescurve でレスポンスカーブを求め、mdpfcalib で波長較正済み 2 次元スペクトルにする手順は同じです。ただし、apall は MEF ファイルを処理できないので一つ目のエクステンションだけを imcopy で取り出します。MEF ファイルを mdpfcalib に読み込ませた場合、前述のようなエラー計算をします。

```
—> imcopy HK500_M53735.fits[0] HK500_M53735_0.fits
—> apall HK500_M53735_0.fits
—> rescurve HK500_M53735_0.ms.fits 8.856 resc_CDFN2_HK500.fits
—> mdpfcalib HK500_MODS11-0390.fits resc_CDFN2_HK500.fits
 HK500_MODS11-0390fl.fits
```

A.2.7 スペクトルの 1 次元化と輝線フィット

ここから下は matplotlib のグラフ窓を使うため、mcsmdp を ipython モードで起動します。

```
$ mcsmdp —ipython
```

mdp2dplot と mdpextract という 2 つのタスクを使います。mdp2dplot は、2 次元スペクトルを引数に指定して、指定した中心波長 (center)、波長範囲 (width) について、nsum ピクセル空間方向に足し合わせて表示します。

mdpextract は、SPECFIT が必要とする輝線の定義データベースファイルを自動的に生成して複数の輝線の同時フィットを行います。輝線の種類は fittype で指定し、今のところ “Halpha” ([NII]-Hα)、 “OIII” ([OIII]-Hβ)、“emission”（単一の輝線）をサポートしています。SPECFIT は対話的なグラフを使ってフィットのパラメータを決めます。SPECFIT のヘルプを参照してください。また、フィットがうまくいくように初期値を決める必要がありますが、これは何度も試してうまくいく値を探してください。

最後に、mdpextract で切り出したファイルをもう一度 mdp2dplot の引数として指定すると、ベストフィットの関数とともにグラフにプロットします。

使用例は以下の通りです。

```
In [1]: mcsmdp
In [2]: unlearn mdp2dplot
In [3]: iraf.mdp2dplot.xlabel=r'$\lambda_{\rm obs} \left[\AA\right]$'
In [4]: iraf.mdp2dplot.ylabel=
 r'$F_\lambda\,[\times10^{-18}\,erg\,cm^{-2}s^{-1}\AA^{-1}]$'
In [5]: iraf.mdp2dplot.bottomplot=False
```

```
In [6]: iraf.mdp2dplot.yscale=1e18
In [7]: iraf.mdp2dplot.linelist="mdpdb$linelist/nebulae_tex.gaia"
In [8]: iraf.mdp2dplot.linewidth=1
In [9]: iraf.mdp2dplot.linealpha=1
In [10]: iraf.mdp2dplot.lstyle='k--'
In [11]: iraf.mdp2dplot.ymin=0
In [12]: mdp2dplot HK500_MODS11-0390fl 59 21990 width=800 nsum=5
In [13]: mdpextract HK500_MODS11-0390fl 59 21990
 HK500_MODS11-0390fl_Halpha width=800 nsum=5 sub_frac=0.5 sub_fwhm=42
In [14]: mdp2dplot HK500_MODS11-0390fl_Halpha 59 21990 plotfit+
 ymin=0 width=800
In [15]: pylab.savefig('HK500_MODS11-0390_fl_Halpha.png')
```

この通りに実行すると図 A.1 のようなプロットが得られます。

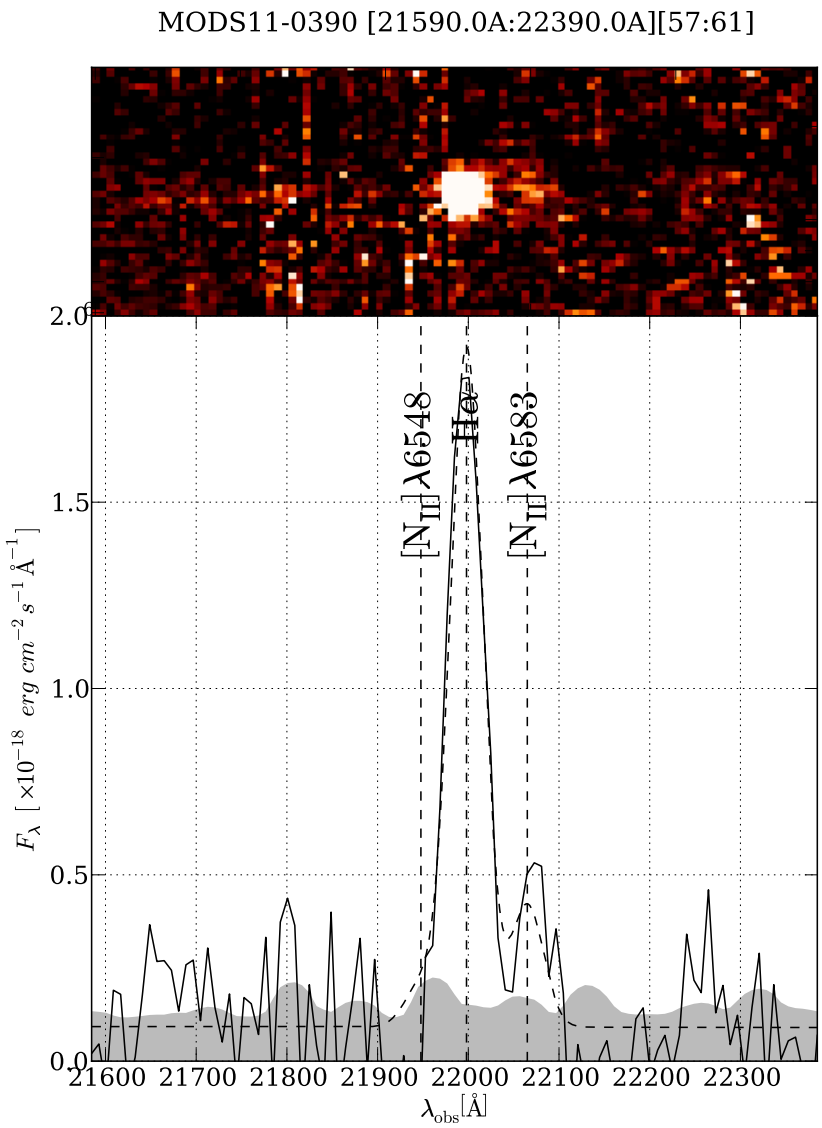

図 A.1: mdp2dplot でプロットした 1 次元スペクトルとフィット結果。