Roque de los Muchachos Observatory

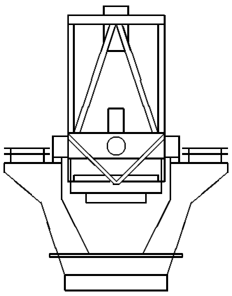
William Herschel Telescope

TAURUS DATA AND HOW TO REDUCE IT

Version 1.0

1991 June 26

Jim Lewis and Steve Unger



The Isaac Newton Group of telescopes on La Palma is owned by the UK Science and Engineering Research Council and operated by the Royal Greenwich Observatory for the SERC in partnership with the Organisatie voor Zuiver Wetenschappelijk Onderzoek (ZWO) of the Netherlands, the Irish National Board for Science and Technology and the Dublin Institute for Advanced Studies.

Contents

1 Introduction: scope of the manual and documentation

1.1 Scope of the manual

This manual describes the use of the TAURUS data reduction package, TAUCAL. Section 1 gives an overview of the manual, and indicates other sources of help. Section 2 provides an overview of the theory of TAURUS data reduction, and section 3 gives an overview of various software issues. Section 4 describes the phase calibration software in detail, and section 5 describes the software for profile analysis (i.e. velocity field mapping). Programme details are summarised in the appendices.

1.2 Obtaining copies of this manual

This manual has been prepared using LaTeX. A copy of the current source is held on the La Palma VAX 8300 in directory `[MANUALS.WHT_TAURUS]`, along with its associated rasterized file, `.DVI-CAN`. New copies of the manual may be printed out using:

- `WHT_GLASE [MANUALS.WHT_TAURUS]TAURUS_RED.DVI-CAN` or
- `INT_GLASE [MANUALS.WHT_TAURUS]TAURUS_RED.DVI-CAN`,

(to give output on the laser-printers in the WHT and INT buildings, respectively). A copy is also held on the STARLINK VAXcluster at RGO Cambridge, in the directory `CAVAD::DISK$USER1:[MANUALS.WHT_TAURUS]`.

1.3 Getting help

The present manual is designed to be the main reference for the use of the TAURUS reduction package, TAUCAL. The operation of the TAURUS intrument at the William Herschel Telescope is described in the *TAURUS Users Manual* by Steve Unger. More specialised advice on TAURUS data reduction can be obtained from Jim Lewis (author of software) or Steve Unger (TAURUS project scientist).

2 General principles

This section describes the basic principles of TAURUS reduction using the TAUCAL package.

It is assumed that the reader is familiar with the basic principles of Fabry-Perot interferometry. If not, it would be advisable to read some of the references listed in section 2.8.

2.1 Basic equations

This section gives the basic equations describing the transmission of a Fabry-Perot etalon when placed in TAURUS. The aim is to set the formal background, before discussing the reduction process. See references in section 2.8 for a more detailoed discussion of this background.

The transmission of a Fabry-Perot etalon is given by equation 1:

$$T \propto \frac{1}{1 + \frac{4N^2}{\pi^2} sin^2(\frac{2\pi\mu G cos\theta}{\lambda})} \tag{1}$$

T is the transmision of the etalon
N is the etalon finesse
μ is the refractive index of air
G is the gap between the two etalon plates
θ is the angle of incidence of light on the etalon
λ is the wavelength

The transmission peaks at wavelengths given by equation 2 (the condition for constructive interference) :

$$n\lambda = 2\mu G cos\theta \tag{2}$$

n is an integer, the order of interference
λ is the wavelength
μ is the refractive index of air
G is the gap between the two etalon plates
θ is the angle of incidence of light on the etalon

The wavelength separation between adjacent transmission maxima (i.e. the separation between successive interference orders) is referred to as the free spectral range. The on-axis ($\theta = 0$) free spectral range is given by equation 3:

$$FSR = \lambda^2/2\mu G = \lambda/n \tag{3}$$

FSR is the free spectral range
λ is the wavelength
μ is the refractive index of air
G is the gap between the two etalon plates
n is an integer, the order of interference

The way in which Fabry-Perot etalons are used in TAURUS is described in detail elsewhere (see section 2.8). Briefly, light is first collimated, then passed through a Fabry-Perot etalon placed in the pupil, and finally reimaged onto a detector. Under these circumstances, the value of θ corresponding to a particular image point on the detector is given by equation 4 (using the small angle approximation for θ):

$$\theta = R/F_{cam} \tag{4}$$

θ is the angle of incidence of light on the etalon
R is the distance of the image point from the point (x_0,y_0)
(x_0,y_0) are the coordinates of the optical axis on the detector
F_{cam} is the camera focal length

2.2 The need for phase calibration

In normal operation of TAURUS, the order of interference is fixed by the use of an order sorting interference filter. Under these circumstances, it can be seen from equations 2 and 4 that at any given point on the detector (i.e. for a particular value of θ), the wavelength transmitted is directly proportional to the etalon gap. By varying the etalon gap, it is possible to change the wavelength transmitted. We can therefore obtain a spectrum for any particular point in the field by scanning the etalon gap, and measuring how the transmitted intensity varies as a function of the gap.

Of course, the aim is to do much more than obtain a spectrum of one point in the field. The principal advantage of TAURUS as compared to a slit spectrograph is that it is possible to obtain spectra simultaneously at every point covered by the detector, by taking a series of images, each with a different value of the etalon gap. These images are stacked together to form a datacube.

The problem with such a datacube is that the transmitted wavelength is not just a function of etalon gap, but is also a function of position on the detector. Thus the raw TAURUS datacube does not have separable spatial and spectral axes, as should be the case for a calibrated spectral line datacube. It is necessary to rebin the datacube, applying a shift to each spectrum so that all spectra have the same wavelength calibration. This shift is referred to as the **phase shift**, the magnitude of the phase shift as a function of position in the image is referred to as the **phase map**, and the process of determining the phase map and rebinning the datacube is referred to as **phase calibration**.

2.3 Functional form of the phase map

From equations 2, 4 and 3 the surfaces of constant wavelength in a TAURUS cube are defined by equation 5:

$$G = \frac{\lambda^2}{2\mu FSR} sec(\frac{R}{F_{cam}}) \tag{5}$$

G is the gap between the two etalon plates
λ is the wavelength
μ is the refractive index of air
FSR is the free spectral range
R is the distance of the image point from the point (x_0,y_0)

(x_0,y_0) are the coordinates of the optical axis on the detector
F_{cam} is the camera focal length

In practice, when we scan the etalon gap, the gap and the free spectral range are not expressed in the same units as the wavelength, but in 'miroprocessor units' (see later). We define a constant a such that $G_m = aG$ and $FSR_m = aFSR$, and can then rewrite equation 5 as equation 6:

$$G_m = \frac{\lambda^2}{2\mu a^2 FSR_m} sec(\frac{R}{F_{cam}}) \tag{6}$$

For small values of R/F_{cam}, sec(R/F_{cam}) can be expanded as a Taylor series to show that the surfaces of constant wavelength are parabaloids of revolution. However, the surfaces of constant wavelength in a spectral line datacube are its planes. Thus each spectrum needs to be shifted by an amount given in microprocessor units by equation 6.

This shift is the **theoretical phase map**. In practice the actual phase map may also include a small contribution from a **residual phase map**, in order to account for effects such as field or detector distortion.

2.4 Determination of the phase map

From equation 6 it can be seen that in order to derive the theoretical phase map we need to know the following parameters:

- Wavelength. This is determined accurately enough for present purposes by the order-sorting filter used. The question of accurate wavelength calibration is discussed separately below.
- Camera focal length. This is a known optical property of the system.
- Pixel size. For a geometrically stable detector such as a CCD this is also simple. However, detectors such as the IPCS, which are not geometrically stable, are more difficult. The normal procedure is to observe a pinhole mask. This is an aperture mask in which a number of holes have been drilled, on a regular grid. If we illuminate this pinhole mask, and take an image, we produce a set of reseau marks on the detector. These can be used to determine the detector pixel size, and can even be used to determine the way in which the pixel size varies across the detector.
- The free spectral range and constant a. This is the difficult bit, and the way in which these parameters are derived is described below.

It should be apparent from equation 6 that if we just look at one plane of the datacube (i.e. $G_m = constant$), then the wavelength is constant on a line defined by $R = constant$. In other words, the wavelength is constant on a ring of radius R centred on the optical axis. Thus if we illuminate the field with a monochromatic light source (e.g. an arc lamp, plus an interference filter to select one arc line) and take an image, we will see a ring centred on the optical axis.

In order to derive the free spectral range and constant a, what we do is obtain a datacube, with TAURUS illuminated by a monochromatic light source. We then fit a ring pattern to the data in each plane of the datacube, in order to determine the centre of the ring pattern (the optical axis) and the ring radius. We then plot sec(R/F_{cam}) as a function of G_m. This plot is referred to as the Radius-versus-Frame (RVF) plot.

What we should see in this plot is a family of straight parallel lines, one line for each order of interference observed. We then determine the gradient and intercept on the G_m axis of each line. The difference between the intercepts of adjacent orders is equal to the free spectral range. We can then derive the constant a from the gradient of the lines, using equation 6

We can now derive the theoretical phase map for any wavelength. In order to determine the residual phase map, we derive the theoretical phase map for a calibration cube, and determine the difference between the observed and theoretical phase surface.

2.5 Flexure and etalon drifts

The theoretical phase map can change as a function of time for two reasons. Firstly, if the optical axis shifts as a result of flexure. Secondly, if the etalon gap drifts.

We correct for these effects by taking a series of 2D exposures through the night at the same nominal gap setting. It is then possible to fit a ring pattern to each image. Changes in the optical axis appear as a shift in the centre of the ring pattern, and changes in the gap appear as a change of the ring radius. The reduction software is used to analyse the variation of ring centre and radius with time, referred to as the RVT relationship.

2.6 Phase correction

We can now determine the theoretical phase map for any specifed wavelength, corrected for effects of flexure and etalon drift. We then add in the residual phase map, and are now ready to correct the data.

The rebinning of the datacube essentially involves sliding each spectrum along by an amount specified in the phase map. The process is somewhat complicated by the fact that the spectra are actually cyclic.

JIM.. could you expand here.

2.7 Wavelength calibration

JIM... could you explain how this works

2.8 Further reading

JIM... there are two recent textbooks specifically about FP interferometers. One is by Vaughan, the other is by Hernandez. There is also an article by Keith Taylor in the book by Roberts called 'Indirect imaging'. Could you dig out the details and add them, cos I can't find them here.

2.8.1 Theory of Fabry-Perot interferometers

Meaburn, J., 1976. Detection and spectrometry of faint light, publ. Reidel.

Atherton, P., 1988. in *Instrumentation for ground-based optical astronomy*, ed. L.B.Robinson, publ. Springer-Verlag, p.124.

2.8.2 TAURUS 1

Taylor, K & Atherton, P.D., 1980. *Mon.Not.R.astr.Soc.*, **191**, 675.

Atherton et al., 1982. *Mon.Not.R.astr.Soc.*, **201**, 661.

2.8.3 TAURUS 2

Unger et al., 1990. *Mon.Not.R.astr.Soc.*, **242**, 242.

Unger, S.W., 1991. *TAURUS users manual*, publ. RGO.

3 Software overview

This section considers a number of general points concerned with TAURUS data reduction using the TAUCAL package.

3.1 FIGARO

Note that FIGARO is environment. Give references.

3.2 NDPROGS

A number of routines, produced at RGO, for analysis of cubes in FIGARO. Reference SUNs for more info.

3.3 Quotas

What quotas are needed to reduce TAURUS data ? VIRTUALPAGECNT etc.

3.4 Sorting cubes

Efficient access implies need to sort cubes. Use NDPROGS transpose

3.5 Datatypes

When to use I2, when to use R4

3.6 Magic values

Are they supported ?

3.7 TAUCAL files

Describe the idea of these here

4 Phase calibration software

4.1 Introduction

In this document the calibration procedure has been broken down to 6 steps:

1. **Creating a 'TAUCAL' file** – This is a bookkeeping device which can be terribly useful. Basically, it gets really tiresome having to type things like wavelength, collimator focal length, etc repeatedly, and using this system cuts down on that sort of rubbish quite a lot. It's also useful as a notebook for results. Quite frequently results from one program are needed in another and storing these results in one 'notebook' is a lot easier than having all sorts of ASCII files floating around with various names which you can never remember. One can run the software without one of these as the TAUCAL system is designed such that if it can't get a certain bit of info from the TAUCAL file (for one reason or another) then the FIGARO parameter system takes over and asks you for it.
2. **Creating a new pixel grid** – What this procedure does is to map out the detector distortions. It then creates a grid of 'pixel numbers' which later programs use instead of the indices to the data array. e.g. for data point DATA(100,200) X = 99.5, Y = 200.04, instead of X = 100, Y = 200. You need a pin hole mask for this. Again, you can run without this as the programs will just use the indices (answer 'no' when they ask you if you want to use a grid).
3. **Do the RVF analysis** – This is the procedure which is used to calculate the free spectral range, the etalon gap and the coordinates of the optical axis from the calibration cube.
4. **Do the RVT analysis** – What this does is to try and take account of changes in the phase map as a function of time. 'Time' here is a fuzzy term as sometimes the crucial factor is not time, but rather whether the telescope has been moved. What you need for this is a series of single calibration rings (preferably one before and one after each observation). The radii and centres of these rings are compared with those of the one ring of the cal cube which was taken with the same etalon spacing as the single rings. It is possible to cope without this.
5. **Create a residual phase map** – Although the taking the IPCS distortions into account in (2) should take out any residual phase, this is probably a good thing to do as at large radii, the rings in the cal cubes can become pretty wimpy and radius estimates become correspondingly worse. You can leave it out though if you want.
6. **Create a final phase map and phase correct** – This is the final step where the actual phase map for each cube is calculated and the cube is phase corrected. The data will then be ready for profile fitting.

Having got all that out of the way, here is an explanation of each program in more detail. They will be addressed in the context of the 6 categories mentioned above. If you are still lost after this 'explanation' try looking at the prologue of the source files by doing **$HELP <program> SOURCE** or leafing through appendix A. Throughout this section the assumption will be made that you are using a TAUCAL file. Not using one doesn't change the way the programs work, but you may then have to answer an awful lot of questions.

4.2 Create a TAUCAL File

CALBASIC This creates the basic TAUCAL file. When you run the program you'll get menus for telescope setup, etalon used and detector used. Just type the appropriate numbers when asked.

If your setup doesn't correspond to any of those offered, then the files to update are *.std where * is SETUPS, DETECTOR or ETALONS. The parameter CALOUT will be the name of your TAUCAL file.

CALRUN This lets you input all the information necessary for each structure. Information about each structure is stored by its run number. There are four different run types:

- CAL – Calibration cubes
- RING – Single rings (for RVT)
- OBS – Observation cubes
- MASK – Pinhole matrix masks.

The first thing for which you are asked is a run type. Since you can enter information about more than one structure at a time, it's best that you do, say, all the obs cubes at once or all the ring frames at once (that way you can take advantage of the default settings for, say, the wavelength).

Once you have given the run type, you are asked for the number of structures of this type and their run numbers. The following information is then needed for each run:

Table 1: Information Required for Each Run Type

Type	*Info needed*
CAL	image name mpu0 (see below) dmpu (see below) wavelength
MASK	image name
RING	image name wavelength
OBS	image name mpu0 (see below) dmpu (see below) wavelength (redshifted)

MPU0 and DMPU are the Z axis zeropoint and increment in microprocessor units (such that the value for the first frame is MPU0). These are useful as there will come a time when the frames are not equally spaced in the spectral direction. Note that the software doesn't cope yet with unequal spacing, but will in the future. Whenever you examine results in the spectral direction it is important to remember that they will almost always be in MPUs. So if, for example you do a 2-d plot of the residual phase map later on and discover that there are many pixels with residuals of 5 or so, this isn't so bad if DMPU is 15, say. Next thing to remember is that all wavelengths are expressed in microns as is the mask distance parameter. As a matter of fact, most if not all physical distances are expressed in microns, even the camera focal length, etc.

If you are interested in what is in the TAUCAL file, you can, of course, always find out by doing **$exam taucal...**

4.3 Create a Pixel Grid

FINDPKS This routine finds the pinhole peaks on the matrix mask frame. It uses a thresholding criterion for locating them and this is, of course, never fool-proof. (e.g. it may miss some or it may include ghost images, etc). So, if you opt to display the frame then you can use the cursor to include any which it has missed or zap any which it shouldn't have included. This might become tedious, so one always has to option of just starting over and using a different threshold as it doesn't take very long anyway.

If you do display the frame, then the cursor routine gives you 5 options:

A: Add a point

Q: Quit

R: Redraw the frame

Z: Zap a point

?: Help menu

When you add a point it should come up in red as an open circle. If you zap a point it should come up as a filled circle. Type 'Q' when you're satisfied with it all.

PEAKDISP If for some reason or another you want to see the peaks again (e.g. to check that you haven't included any ghosts) use this program.

TMASK This routine does the 2-d polynomial fit to the peaks' positions. From the number of peaks it tells you what the maximum order it's willing to fit is. It calculates a rough rotation and then does a proper fit. The file MASK.OUT is interesting because it gives the pixel sizes in both directions and the pixel ellipticities all as a function of position. It's useful to check to see if the pixel sizes are on the same track as what you would expect.

SIZES If you really want to look at a map of pixel sizes, you can generate one using this program. It asks for the size of the map you want and for the axis you want to see. The output is a 2-d image which you can display with the NDPROGS routine DEPICT or whatever you want.

GRID This actually creates the new pixel grid. This grid can be of any size, although one should worry about extrapolating the fits, of course. The reason this size flexibility has been included is because there are times when some cubes cover the whole detector and others in the same observing run are a subset. Thus, this was intended to do subsetting. Basically you are asked if the map is to be of a different size in X,Y from the matrix file and if so, what size in each direction. You are then asked if the new grid is to have the same physical centre as the matrix file. If not, then it asks where you want to centre the new grid.

4.4 Do the RVF Analysis

PRERING This is the program which gives the ring fitting program (RINGS) the initial estimate of the ring parameters. The first thing to do is to run the NDPROGS routine MOVIE on the CAL cube to find out where each series of rings begins and ends. Basically what happens here is that you are asked to show where the ring is on the frame for 2 rings for each series. The program fits a circle to the points you have indicated. The program then extracts data along a radius from the centre of the circle through each of your indicated points. The data are added together and averaged and a gaussian is fitted to it. This gives a better estimate of the width. Then the program writes out a file of what it thinks is a best guess for the parameters of each ring. In doing so it uses the following rules:

$$R^2 = AZ + B$$

where Z is frame number and R is radius, and

$$R\sigma = \text{const}$$

where σ is the width of the ring. To get a good guess at A and B, one should make sure that the rings used in each series are a reasonable number of frames apart (say 5–10) if possible. Try to avoid rings with small radii as the small radius rings definitely look more Cauchy–like.

When you are asked if the output is to be an RVF, answer 'Y' as PRERING is also used for estimates for the RVT and the output files are slightly different. NSER is the number of series of rings. (This is usually 2 but you can have more. At the moment, the routine which calculates the FSR doesn't cope with series of different wavelengths, if, for example, your calibration source is a doublet. So for the moment, use the rings of one particular wavelength and stick with it.) When you're asked for SER_NOS, you need to input 2 values which are basically the first frame of the series and the last frame of the series. Obviously, if you have a frame where there is only a teeny part of a ring in one corner, it isn't worthwhile to try and fit it, so be judicious if you don't want to waste time. FRAMES_PROC are the 2 frames from each series which you will be using to define the parameters for that series. If, for example, the first series spans frames 1 to 31, then try processing frames 1 and 10. After asking you for the name of the output file, the program should then do a 2-d colour plot of the first ring. You can indicate with the cursor up to 10 points on the ring. Obviously the more the better as it's advantageous to make sure you indicate enough points around the ring to give the routine an idea of the curvature, otherwise the answer will be garbage. Hit the spacebar to identify a point. If you want to quit identifying points, type 'Q'. The fitted circle should then appear on the plot. The plot should then erase and a line plot of the profiles with a gaussian fit should appear. This procedure repeats $2 * \mathrm{NSER}$ times. It may sound a bit tedious, but in reality only takes about a minute.

CRE_RV The output file from PRERING is used as both the input and output from the ring fitting program (RINGS). If you have somehow screwed up this file (e.g. deleted it, run RINGS with the wrong parameters, etc.), it is possible to recover all of the info from PRERING (if you are using a TAUCAL file), by using this program.

RINGS This is the program which does the actual ring fitting. It's fairly uncomplicated in the way that it works. If the ring frames are fairly large, quite a bit of the information is somewhat redundant. So if you want your answers more quickly, then the parameter COMP_FAC may be of some help. This governs what fraction of the pixels which the program actually takes into account. e.g. if COMP_FAC = 2 then every second row and every second column is used or every FOURTH pixel. One should be fairly careful here though as the rings with large radii are fairly thin and if one is cutting out a large percentage of the pixels, it may be that you don't give the program enough to go on. TOLCENT and TOLRAD are what you consider to be a threshold of pain for the errors in the position of the centre (X_c, Y_c) and the radius R. If the error is greater than these tolerances, then the fit is flagged as bad. WEIGHT is a flag for the weighting scheme you want to use. At present there are $\sqrt{n}$, or equal weights available. You are asked whether you want to use a grid as described in the previous section. RESTRICT allows you to restrict the area to be used in fitting to a certain number of sigma around the ring. This is useful if you have multiple rings on a frame. Running this program in batch may be a good thing especially for large frames where COMP_FAC = 1 and RESTRICT = 'N'.

PRINT_RV This is a program to take your RVF or RVT file and make an ASCII copy of it. The resulting file is *file*.OUT where *file* is the name of our RVF or RVT. This is useful in case you think you've got something wrong, either with PRERING or RINGS.

EXAM_RVF This is the program which takes the RVF file from RINGS and calculates the FSR, the gap and the mean centre of the rings. Note that you MUST have access to some sort of soft graphics device for this program. The only parameter which may need some comment is SAME_SLOPE. What this does is allows you to constrain all the slopes to be the same. This is useful if you have one good series and one bad one (for whatever reason). If you answer 'Y' to this, then you will have to nominate one of the series to be the 'fiducial' (i.e. 'good') one. The rest is straight forward. On the graph, you will get points plotted either as open or filled circles. Hollow circles are the rings which fit properly in RINGS and filled ones are ones which either didn't fit well or you skipped when you nominated the endpoints for each series in PRERING. Occasionally one will appear as an X, which means that RINGS thought the fit was bad, but not too bad; those rings will get half weight. All series which you mark out in the current version of this program must be of the same wavelength. N.B. *the program expects you to mark out the series from left to right.* For example, the left-most series is series #1. If you use a fiducial series then you will be asked to mark that one out first. Each succeeding series should then be done starting from the left most series. Note that the fiducial series is only to be marked once. When you are asked to mark the endpoint of each series with the cursor you have three cursor options:

Q: Quit marking out series

S: Skip the current series

Anything else: Mark the series endpoint at that point.

You can easily skip ones on the ends which have a dodgy fit by just skipping over when indicating the series endpoints. Don't worry if there appears to be a dodgy fit somewhere in the midst of a lot of good ones – you can always delete it later. You are then asked if you want to fit both the full and half-weighted points or just the full weighted ones. A fitted line should appear. If you then want to delete any of the interior points or change the weight of a point, you can. You will then have the following options:

F: Full weight a point

H: Half weight a point

Q: Quit editing the fit

R: Redraw the plot

Z: Zero weight a point

?: Print up a help message

You basically need to repeat the procedure of indicating series and editing fits until you have fit all the series you want. Note that you need at least two series to calculate both the FSR and the gap. If however you only have one series in your data, but you have a good indication of the FSR from elsewhere, the program will then ask you for this estimate and it will give you an estimate of the gap. X_c and Y_c are calculated as the mean centre of all of the rings used in this program. Note that the FSR is in MPUs as are the regression coefficients.

4.5 Do the RVT Analysis

PRERING One uses this program pretty much as you did before with a few of exceptions. First, when it asks if the output is to be an RVF answer 'N'. If you are using a TAUCAL file, it will tell you how many ring frames are available for fitting and ask if you want to use them all. If you don't want to use them all, it prompts you for the rings to be used. If you are not using a TAUCAL file, then you are asked to supply the names of each of the individual ring structures. In one way or another you will also need to supply the program with a frame from the CAL cube. This frame should be the one which was taken at the same etalon spacing as the single rings (as is given from the microprocessor coordinates). A 2-d plot of the first ring will appear, you will be requested to indicate points on the ring and you will get the profile fit as before. For the rest of the rings, the program takes the first estimate of the ring centre and extracts profiles from the other rings along the radial paths between that centre and the points you indicated on the 2-d plot. You should then get a series of line plots with profile fits for each of the rest of the rings in the RVT.

RINGS Again, you use this in a very similar way as before. When it asks if this is to be an RVF then answer 'N'. The program will open the series of ring files and will do all the fits at once.

4.6 Create Residual Phase Map

EXAM_RVT This program calculates offsets in X_c, Y_c and$Z$ which may be due to drift over time or from slewing of the telescope. You have a choice of 4 estimation methods.

- Use the X,Y,R from the ring taken just before the OBS cube
- Use the X,Y,R from the ring taken just after the OBS cube
- Use an average of these from the two adjacent rings
- Use an interpolation between the two.

Which one of these you choose is up to you. The plots you get are all self-explanatory. It may seem a little silly to you to do this for the CAL cube, which is what is being suggested here. After all, δR will equal 0 and hence so will δz. However the X_c and Y_c for the CAL cube ring used for comparison *may* (and in general *will*) not be the same as the average for the whole cube and thus the values of δx and δy may make the residual phase map more valid.

RESIDUAL This program takes the FSR, gap and regression coefficients from EXAM_RVF and predicts where the emission line should occur in the CAL cube as a function of (x, y) position. It then does a moments analysis and calculates the residual. The only parameters likely to confuse are: WINDOW which is a window in the spectral direction over which to search for the line. This helps to avoid noise spikes and brighter lines in the case of multiple rings. A reasonable value here is 3. SMOOTH_SIZE is a smoothing box in the x,y plane, and really helps out in the larger radius pixels where the rings start getting very weak. It's usually a good idea to examine the output map on a 2-d plot, remembering that the result will be in MPUs!

4.7 Create the Phase Map and Phase Correct

(***Note that phase correcting the CAL cube is a good way to check that you've got all this right. Thus from here on out the term 'OBS cube' can also refer to the CAL cube.***)

EXAM_RVT This program calculates offsets in X_c, Y_c and$Z$ which may be due to drift over time or from slewing of the telescope. See section 4.6 for more information.

LAMDA This program does the wavelength scaling of the OBS cube. Actually you can do it for a number of OBS cubes at a time. The information which appears on the screen is very useful and most of the important bits are stored away in the TAUCAL file, if you are using one.

PHASE_MAP This program calculates the 'phase map', that is a map which tells the phase correction program how many MPUs (pixels) to shift each spectrum (in the negative direction) so that the desired wavelength appears at the central spectral pixel. It doesn't ask any difficult questions.

CORRECT Note that before you can use this program, the OBS cube MUST by ZXY sorted, (i.e. the spectral axis must be the first axis of the data cube– the NDPROGS routine TRANSPOSE is useful for this). As a result of this, the program asks you for the name of the file rather than reading it straight from your TAUCAL file. This is the program which does the actual phase correction. You are given the choice to do either a linear interpolation or a sinc interpolation. This choice of interpolation scheme can be problematic. For certain shifts near about 0.5 the sinc interpolation weights go roughly like so: -0.1 0.6 0.55 -0.05, or something similar. As you can see, they all add up to 1 which is right, but something funny happens if you use INTEGER*2 data. If you consider the following program:

```
program test
integer*2 data_old(10),data_new(10)
real blah,w(5)
data data_old /0,0,0,0,1,0,0,0,0,0/
w(1) = 0.0
w(2) = -0.11
w(3) = 0.56
w(4) = 0.69
w(5) = -0.14
do j = 3,8
   blah = 0.0
   do i = 1,5
      blah = blah + float(data_old(j+i-3))*w(i)
   end do
   data_new(j) = nint(blah)
   write(6,*) blah
end do
end
```

ΛΛ The weights here are the sinc function weights for a shift of 0.55. If you run this program, you'll find that 'blah' becomes:-0.14,0.69,0.56,-0.11,0,0. Hence data_new(3:8) become 0,1,1,0,0,0. In other words, it doesn't conserve flux. What happens in this case is if you use the sinc interpolation like this on INTEGER*2 data and then you collapse the resulting cube in the spectral direction, you see a ring pattern in the background which corresponds to the places in the phase map where the fractional shift is near 0.5. This obviously isn't a problem for real data. Note also that if the data are converted to integer by doing 'int' as opposed to 'nint', you still wouldn't conserve flux as then data_new(3:8) would become 0,0,0,0,0,0. Hence for INTEGER*2 data it is recommended that you use the linear interpolation scheme, rather than the sinc interpolation. If this bothers you, then convert your INTEGER*2 data to FLOAT using the NDPROGS routine TYPECON. For real data, it doesn't matter which you use.

The rest of the parameters are all easy and the output file will also be ZXY sorted. By the end of this step you should have a phase calibrated cube which is ready for profile fitting.

5 Profile analysis

5.1 Introduction

In order to get some science out of this enormous calibrated cube you now have, it will be necessary to do some sort of analysis of the individual profiles. This can be accomplished in a number of different ways:

Moments analysis – This is probably the quickest way to get the results you want. It has the advantage over profile fitting of being a relatively objective way of parameterising the data.

However, there are major disadvantages. Unless your spectra are noise free, you will always get a somewhat biassed result from a moments analysis. The results can also be very noisy if the threshold is not set correctly. Still, for very high signal/noise data, this may not be a problem, and moments analysis can be a useful way of getting a quick first look at the velocity field. If this is the route you wish to take, then the NDPROGS routine MOMENTS is the program for you.

Cross correlation – This has the advantage of using all of the information in each spectrum. It is therefore a good way of getting the velocity field when a large number of lines are present. It is reasonably objective, though there may be a problem defining a sensible template spectrum to cross-correlate with.

It is, however, very time consuming and in the long run something like a profile fit has to be done in order to measure the peak of the cross correlation function. This method also has the disadvantage that at the time of writing this document, no routine to do this exists. You could always write your own...

Profile fitting – The time taken to analyse a datacube using profile-fitting depends upon how fancy you want to get with your fitting function. There are times when several components can be seen in the data and in this case a multi-component profile fit is the only way to extract the data. This can be rather time-consuming. There are also cases where the line profile is too complex to be modelled in terms of any simple set of parameters.

However, for the reasonably simple spectra that one is most likely to encounter with TAURUS data, a fast and efficient routine (TAU_FITS) has been developed. The use of this program is the subject of the rest of this section.

5.2 Introduction to TAU_FITS

5.2.1 Preparation of the Input File

Before you start doing any line fitting there are one or two things you might consider doing to your calibrated cube. First of all, there are a number of things which TAU_FITS needs to know. These include the free spectral range, the wavelength zeropoint and increment etc. TAU_FITS will either read these from the header of your cube or will ask you for them directly. If you want to avoid being asked for this sort of information all the time and you think it would be a good idea to have this sort of information in the header of your cube (which it is, for general reference), then the program TAU_HEAD is very useful. If you have used a TAUCAL file, then it just takes the information straight from there and places it in the header of the cube. If you haven't you'll be prompted for the information. A substructure called *file*.MORE.TAURUS is created with all the information.

TAU_FITS also maps input files as REAL. What this means is that if you use an INTEGER*2 cube as input, then TAU_FITS has to spend quite a bit of time converting the data internally before any fits can be done. This was done quite deliberately as trying to fit to integer data will not yield great results owing to rounding errors. Thus converting your cube to REAL (using the NDPROGS routine TYPECON) will save a lot of time in the long run. Doing a gentle smooth on the data will not only help the signal/noise, but will also smooth away those nasty round off errors which you got from CORRECT if you insisted on INTEGER*2 output. The NDPROGS routine SMOOTH allows you to smooth data in up to 3 dimensions using a number of different smoothing functions. The run time depends on the size of the smoothing function (e.g. the σ of a gaussian) and can be rather slow for large widths. It's probably best, therefore to run this in batch overnight.

5.2.2 The Main Menu

When you first enter TAU_FITS you will be asked for a number of things, none of which is particularly difficult to understand. You will then get a message which tells you what the default parameters are currently and a prompt for an option. If you type 'H' then you get the help message which explains all of the options available in the main menu. Before the guts of the program are explained, it will probably be worthwhile to explain some of the logic behind the program.

TAU_FITS tries its best not to carry around more data than is absolutely necessary. This is based on the problem of finding enough page file for the large bits of workspace needed to do fits on such a large number of spectra as you usually get from TAURUS. Hence, any 'action' taken on the cube has to be 'limited', e.g. you have to tell the program which region of x,y space you want to consider when taking a certain action, whether it be fitting some spectra, editing some fits or what ever. It is possible to limit the region in three ways:

1. Give the program limits in x,y over which the action is to occur.
2. Give the program a mask image and thresholds between which the action is to occur.
3. A combination of (1) and (2).

It is important to remember to do this, since if you forget, the program may end up converting, binning, etc a lot of data which either you don't want or for which your particular VAX may not have the page file space (the latter of which, of course, will cause the program a major upset).

That warning now over, the first two options to be looked at are MASK and LIMITS. Note that all options in TAU_FITS can be abbreviated to a single character. MASK basically sets a flag which tells the program that you want to use a mask image. It doesn't actually ask for the name of the image or the thresholds. LIMITS actually sets the limits on the spectra to be used. If the mask flag has been set, it will first ask for a MASK image and for some threshold values. Note that the MASK image must have the same x,y dimensions as the cube! Next, LIMITS asks for the minimum and maximum values of x and y which you want to consider. Once you have input these, then limits decides which spectra will be used. To be included, the spectra must meet the following criteria:

$$x_{min} \leq x \leq x_{max},\ y_{min} \leq y \leq y_{max}$$

and given that a MASK image is being used:

$$m_{min} \leq m(x,y) \leq m_{max}$$

where $m(x,y)$ is the value of the mask image at (x,y). If you suddenly decide that you don't want to use a MASK anymore, you can just run the MASK option again and switch it off. However, you then must remember to run LIMITS again.

The next option which you may find useful is PICTURE. This does a 2-d colour plot of an image. Thus, you may have a compacted image of your cube (a 2-d image created by crushing your data cube in the spectral direction – very useful! The NDPROGS routine COLLAPSE will do this for you.) which you want to display. PICTURE asks you for the name of the file, the look up table (LUT) you want and for the min and max for the colour plot. If you run PICTURE for a second time, then you will be asked if you want to plot the same file as before. This cuts down on the amount of time you have to spend opening and closing files, which can become pretty boring at the best of times. The real beauty of PICTURE is that you can use it to help you set the (x, y) limits. If TAU_FITS sees that you have a picture displayed when you enter LIMITS, then it will give you the option to indicate the (x, y) limits with the cursor. This can be very useful as it isn't often that someone knows off the top of his/her head where the object is located on an image! Other uses for the picture will become apparent as time goes by.

If you have low s/n spectra you may want to bin the data. In this case the option BOXSIZE is useful. This defines the size of the binning box you want to use. Note that the box is always square.

TAU_FITS uses a χ^2 criterion to define when a fit has converged. Basically the fit is said to have converged when χ^2 decreases by a 'trivial' fraction n times, where the definition of 'trivial' and n are given by the user. In practice very small fractional changes in χ^2 are never very meaningful, so you won't gain much by setting the threshold too low and you may slow things down considerably. The parameter which defines this threshold is called DCHI and is the definition of a 'trivial' *fractional* change in χ^2. Something like 0.05 is of the right order. The parameter NTRIV tells the program how many times χ^2 has to change by DCHI in order for the fit to have converged. A value of 2 or 3 is usually sufficient – you will not gain much by asking for anything more strict. Finally, you must tell TAU_FITS what is the maximum number of iterations you are willing to do on a given spectrum (30 or 40 usually suffices). These parameters are set using the option CONVERGE.

The option TYPE tells you what fitting functions can be used on the spectra. At the time of writing of this document there are only two choices: a single Gaussian or a single Cauchy function. Both of these assume a constant background.

Again at the time of writing, TAU_FITS doesn't deal with error arrays within the data cube. So you have to tell it how to weight the data in the fit. At present there are two options: equal weights or $\sqrt{n}$ weights. This is set using the option WEIGHTS.

In order to get a fit in a reasonable amount of time, TAU_FITS needs a first guess at the fit parameters. It currently gets this in a number of ways. The first way is to search for the peak and do a moments analysis about it. The second way is just to use an old result (in the case that maybe you want to refine some fits) for that pixel. A third way is to show TAU_FITS with a cursor where the line is. There are two versions of this guessing algorithm. The slow version asks you also to indicate the background level in the spectrum. This method will be discussed in more detail later on. It is not advisable to use this method if you have a lot of spectra to fit, for reasons of dull tedium. Finally one can estimate the fit parameters for one spectrum by using the fit parameters of a nearby spectrum. This option is not available in the main menu and will be discussed later. The option GUESS is used to decide which guessing algorithm is appropriate.

It is worth noting that TAU_FITS doesn't explicitly ask you for these values unless it can't find anything worthwhile in your VARS.DST file. If you have run TAU_FITS once, then your VARS.DST file will retain the last values for all of the parameters described above. Thus, the next time you run it, TAU_FITS will pick up the values for the options: TYPE, WEIGHTS, GUESS, BOXSIZE and CONVERGE. Just before you are asked for the first option you will be presented with a list of these set up parameters. If you then wish to change any of these, you may do so. If at some stage you can't remember what you have used for any of these parameters, then the option INFO will put this same list one the screen again.

Finally the option FIT will do the actual fitting of the spectra. This option gives you a running commentary on how many spectra have been fit out of the total. If it seems to take a long time to do the first hundred, then it is worth remembering that before the fitting routine actually starts, a routine is called which actually transfers and bins the data into a workspace, which can take a significant amount of time, depending upon how may spectra you've asked for in LIMITS and the binning box size. Once this is done, however, the fitting is quite rapid.

The option QUIT gets you out of TAU_FITS completely.

5.2.3 Edit mode

Once you have done some fits, it is worth looking at the results to see if they make any sense at all. This may seem tedious, but is worth it in the long run. (It's also a bit stupid to do a fit and not check it, as even the most sophisticated fitting algorithms are capable of turning out rubbish at times, and human eyes are far better at detecting this than even the most neural program...) Once you have received the prompt in the main menu, you can then edit the fits with the command EDIT. A few things are worth remembering about EDIT before you get started.

1. You MUST have access to a graphics device to use EDIT.
2. EDIT will only edit the spectra which have been defined by LIMITS. If you only want to look a subset of the spectra you have fit, then use LIMITS again.
3. If you change the limits before running EDIT, then the data will have to be prepared over again. Thus there may be a delay before you get the first panel of spectra.
4. Once the data have been prepared, you will be asked for the number of plots to be done on a side. This number squared will be the maximum number of plots on a given panel.
5. Each spectrum will be plotted up in turn starting with the lower left hand corner of the box defined in LIMITS. The fits which have been done will be overlaid in green. There may be instances where you get less than the maximum number of plots on a panel. This is because the total number of spectra on a side of the box defined by LIMITS may not be divisible evenly by the number of spectra on the side of a plot panel.
6. If you have used a mask image, then even though a pixel may be within the (x, y) bounds defined by LIMITS, it may not be within the mask thresholds which you have defined. Those spectra will be plotted as an empty box.

Once you are in EDIT mode the command HELP will give you the list of commands available. (Again all commands here can be abbreviated to one character.) The first thing you'll notice is that there are a number of commands in EDIT mode which are the same as ones which have been described in the main menu (in some cases they are not *exactly* the same, but these cases should become obvious). These are CONVERGE, FIT, GUESS, HELP, INFO, QUIT and TYPE. The commands SHOW and ZAP are also very similar to those in the main menu and will be discussed in due course. You can indicate which spectrum you want to take action on with the cursor and indicate which action by typing the appropriate letter. (If you type a letter which has no corresponding command, then nothing happens.)

If a spectrum has a bad fit, or if it doesn't have much of a line and the program has attempted to fit something to it, then you can get rid of the fit with the option ZAP. This should erase the fit. (Unfortunately this sometimes also erases part of the spectrum, so if you really get confused, the option REDRAW will redraw the current panel). One thing to remember about ZAP is that once something have been ZAPped, it can't be unZAPped without refitting. If you hate all of the spectra on the current panel then the command WIPE will get rid of all of the fits on that panel and will redraw that panel without the fits.

The command SHOW prints up the fit parameters for the indicated spectrum. This will be in the order: peak, λ_o, σ, background, where the peak is defined as the intensity *over* the background. The error estimates for each parameter will be printed underneath.

The GUESS command is exactly the same as before with one exception. You now have a fifth option which allows you to give the fit parameters of one spectrum as the first guess for another.

The command FIT will attempt to fit the spectrum which is indicated by the cursor. If you have chosen the moments or the previous results guessing algorithm, then the fit which it has found should appear straight away (assuming it's found one!). If you have chosen either of the two 'guess by hand' algorithms, you will be prompted to indicate various locations on the spectrum. To indicate a point, use the space bar and the cursor. In the slow version, you can indicate up to 10 continuum points. (If you wish to indicate less then either type 'Q' or indicate a point outside the box – but you *must* indicate at least one!) You are then requested to indicate one point on either side of the base of the line and finally one point at the peak. In the fast version, you are only requested to indicate the sides of the line and the peak; it assumes that continuum is at the same level as the base of the line. If you choose the option to use the parameters of another fit as the first guess at the fit for the indicated spectrum, then you will be asked to indicate another spectrum on the panel whose fit can be used as a template. If you suddenly discover that you don't have any good fits on that panel, then type 'Q' and choose another guessing algorithm.

The command NEXT will then advance you to the next panel of spectra. The command PREVIOUS will move you to the previous panel. If you have decided to quit editing then use the command QUIT. You will be asked if you are serious about quitting. This is to help overcome typing 'Q' by mistake. If you answer YES, then you are returned to the main menu. As you are editing, the coordinates of the spectrum on the lower left hand side of the panel will be displayed on the screen. These may prove useful for setting limits for another editing session.

5.2.4 Nearest Neighbour Fitting

In the main menu, there is an alternative mode of fitting spectra from FIT. This is the option NEIGHBOUR. This mode sweeps through the region defined by limits looking for pixels which have a neighbour with a good fit. If a pixel does have a neighbour with a good fit ('neighbour' is defined as a pixel which touches another pixel either on a side or at the corner), then that neighbour's fit is used as the first guess for that pixel's fit. This has the advantage that the results tend to reflect the smooth nature of the velocity fields of most astronomical objects. The user tells the program the maximum number of sweeps to be taken through the region. (If all of the spectra which can be fit are fit in fewer sweeps than this maximum then the fitting routine stops, obviously). The user also gives the program limits in the parameters. Outside these limits, the fit is judged to be unphysical. This is important as, if an unphysical solution is found for one pixel, then it can propagate across the field becoming more and more ridiculous.

To use this routine, there must be some pixels with good fits (e.g. seed points). This can be done either with FIT or with EDIT, although the latter is favoured as one can then see that the fit is not spurious straight away. To do this one could adopt the following method:

1. Do a 2-d plot of the field using PICTURE and decide from that what is the region you ultimately want to fit.
2. Use LIMITS to delimit a small (circa 50 spectra) region where you think you'll find good spectra.

3. Use EDIT and fit a few of these.
4. Do another 2-d plot.
5. Go back to (2). Do this until you think you have a representative sample across the object.
6. Use LIMITS (and MASK if you want) to define the whole region to be fit ultimately.
7. Use NEIGHBOUR.

If NEIGHBOUR sees that you have a 2-d plot on the screen, then it will overlay a point on pixels where it has tried to do fits, so you can monitor your progress. It also gives you a news report on each sweep. Again, it may take a while before you get your first news report as the routine which prepares and bins the data has to be called before the fitting gets done.

5.2.5 'Cleaning' Velocity Maps

At some stage you might think it wise to actually get some velocity information out of your fits. The trouble is that the wrap around inherent in TAURUS data sometimes causes a line to appear twice on a spectrum. If one is using some sort of guessing algorithm which gets its initial guess for λ_o from the maximum in the spectrum (as the moments algorithm does) then the velocity you get for a particular pixel may in fact be a free spectral range out from the expected velocity. Judicious guessing in the initial stages of a NEIGHBOUR analysis will probably eliminate most of the problems of this kind. But problems may occur if, for example, you fit a large number of spectra in EDIT mode using the moments guessing algorithm. In this case, if two lines appear on the spectrum, then that guessing algorithm will always choose the stronger line.

To get around this problem, there is the option VELOCITY. You *MUST* have access to a 2-d display to use this option! If you already have a velocity map (for example, you've created one using XTRACT (see section 5.2.6) or you've used VELOCITY before and you want to do further 'cleaning') then you can use it. If you don't then VELOCITY will create one from the results structure. You are then presented with a 2-d colour map of the velocity field. If you decide you want to 'clean' this map, you are asked to choose a pixel on the map which has the 'right' velocity (e.g. in the right Fabry-Perot order). You will also be asked for a tolerance (TOL) to be used by the routine to decide whether or not a pixel is an FSR out or not. This is expressed as a *percentage* of the FSR. What happens next is a lot like NEIGHBOUR. You are asked for a number of sweeps. On each sweep, the routine checks each pixel to see if there is a pixel with a velocity in the right order next to it. (Obviously on the first sweep there will only be one such pixel since you just nominated it with the cursor!) If it has such a neighbour, then it checks to see if the two velocities are an integer number of FSRs apart ($\pm$ TOL*FSR). If so, then the 'program' pixel's velocity is changed and it becomes a 'fiducial' pixel. If not, then that pixel is in the 'right' order and is also flagged as a fiducial pixel. If a pixel has no neighbour with a 'good' velocity, then it has to wait until it does. The routine keeps iterating until either it has done the number of sweeps you have given it, or until each pixel with a neighbouring 'right' velocity has been tested.

You will receive a news bulletin after each sweep has been done to tell you how many of the pixels have had their velocities switched. If the first pixel you indicate with the cursor is somewhere in the middle of a sea of other pixels with velocities in the same order, then it may be a while before you see any velocities change.

Obviously if there are several objects on your frame, then you may have to repeat this procedure several times. Thus you will first get a new 2-d plot of the map with the new velocities included. You are then asked if you want to accept the map with these velocities. If you don't accept the changes you are then presented with the old map again. You can finally choose whether or not to repeat this procedure on other objects in the field. Once you have finally decided to finish 'cleaning' the map the routine checks to see if any changes have been made to the velocity field. If there have, then you are given the option to write out the new velocity field to a file.

Note that VELOCITY *does not* change the position λ_o of the fits. It only writes out a velocity map (if you so choose). To do the former would be extremely naughty without doing a refit of the affected spectra. Also, it may be that the line in question has run off the red end (for example) of the spectrum because the velocity field *really does* extend into such a wavelength region which is not covered in your cube. In such a case a line of a lower (by 1 FSR) wavelength may have been fit, but the velocity at that pixel is *clearly* one FSR higher. Thus such a cleaning procedure is vital.

5.2.6 Other Options in Main Menu

There are a few more useful options in the main menu. First there is SHOW, which, as in EDIT mode, will display the fit parameters for a given pixel. You may either type the coordinates of the pixel you want or, if you have a 2-d plot displayed, you may indicate the pixel with the cursor.

The option XTRACT extracts results from the output file which can then be plotted using PICTURE. You can extract any of the results parameters, their errors, their relative errors or the velocity field.

Should you desire all line plots to have the same scale in the Y direction then the option YSCALE takes care of that. You will be asked, first of all, if you want to scale all the plots similarly. If so, you will then be prompted for the ranges to be used. Note, this option can also be used to *switch off* this scaling if you so wish. The default is that this option is off and thus all plots will be scaled in such a way that they all fit.

The final option, ZAP, allows you to reject fits in bulk. This works in several ways.

1. Reject based on (x, y) position
2. Reject based on the output parameters (e.g. all negative intensities)
3. A combination of (1) and (2).

If you are rejecting by (x, y) position you may type the (x, y) limits or indicate them with the cursor (if you already have a 2-d plot up). You are then given the option to zap fits either inside or outside the box you've just defined. If you are rejecting by the results, you will be asked for the ranges for each parameter, outside which the fits will be rejected. This routine can be rather drastic as a ZAPped fit cannot be recovered without refitting!

A complete list of the main menu and edit menu options can be found in appendix B and appendix C respectively.

5.2.7 TAU_FITS in Batch

For reasons of self preservation, it is often a good thing to do large amounts of fitting in batch. Unfortunately, as TAU_FITS is a menu driven program, it isn't possible to run it in batch in the same way as you would interactively. For one thing, it is not possible to fit two different regions in (x, y) space in the same job. This is because the OPTION parameter would have to take on several different values during the course of the run and that isn't allowed by the FIGARO parameter system when a program is running in batch.

To run TAU_FITS in batch, therefore, you have to break up your actions into separate jobs. For instance, if you want to fit two separate regions of a cube, then you should set up two batch jobs to do it. In batch, TAU_FITS needs for things to either be specified on the command line or in your BVARS.DST file. The following table is a list of the parameters for which TAU_FITS will look.

Table 2: Parameters Required for TAU_FITS in Batch

Parameter	*Description*
NAME_OBS	Input data cube
NEW_FILE	Flag for creation of a new output file
NAME_FIT	Output file
ALAM0	Wavelength zeropoint
DLAM	Wavelength increment
FSR	Free spectral range
BOXSIZE	Smoothing box size
NTRIV	Number of trivial changes in χ^2
DCHI	Maximum fractional change in χ^2
NITER	Maximum number of iterations
TYPE_FIT	Fitting function
START	(x, y) minima for LIMITS
END	(x, y) maxima for LIMITS
WT_SCHEME	Weighting scheme
USE_MASK	Flag to use mask image
NAME_MASK	Name of mask image
MASK_THRESH	Lower and upper thresholds in mask image
BATCH_FIT	See below
NSWEEP	Number of sweeps for neighbour fitting
RNG_AMP	Min and max range of peak
RNG_POS	Min and max range of position
RNG_SIG	Min and max range of width
RNG_BAC	Min and max range of background
HOW_GUESS	Guessing algorithm to be used

The parameter BATCH_FIT tells the program whether you want to use the option FIT (BATCH_FIT=1) or NEIGHBOUR (BATCH_FIT=2). The parameters: NAME_OBS, NEW_FILE, NAME_FIT, BOXSIZE, BATCH_FIT, NTRIV, DCHI, NITER, WT_SCHEME, START, END and TYPE_FIT are all required. The parameters ALAM0, DLAM and FSR are required if this information is not in the header of the input data cube. If one of these is not specified on the command line, then the program will look for it in BVARS.DST. If it isn't there either, then the program quits. If the parameter USE_MASK isn't specified on the command line, then the program assumes it's NO. The parameters NAME_MASK and MASK_THRESH are only required if USE_MASK=Y. If BATCH_FIT=1 (FIT) then the parameter HOW_GUESS must be specified. (Note that in batch HOW_GUESS can only have values of 1 or 2, the other options not being possible in batch.) If BATCH_FIT=2 (NEIGHBOUR) then the parameter NSWEEP and all of the RNG_ parameters must be specified.

A typical .COM file to run TAU_FITS in batch might look like this:

```
$set verify
$set def tauvax_sys:[taurus.data.swu]
$figaro
$tau_fits name_obs=n1068 new_file=n name_fit=n1068_fits -
 boxsize=5 ntriv=2 dchi=0.05 niter=30 type_fit=1 -
 wt_scheme=1 use_mask=n start=(133,171) end=(166,206) -
 batch_fit=2 nsweep=100 rng_amp=(0.1,1.0e6) -
 rng_pos=(5019.0,5032.0) rng_sig=(0.14,10.0) -
 rng_bac=(0.0,10.0)
```

In this case, the user wanted to do a NEIGHBOUR fit. In order to do that he/she would have to have had a few good fits in the output file which could have been achieved by running TAU_FITS interactively in EDIT mode and fitting a few spectra by hand. The rest of the frame could then be done in batch.

5.2.8 The Output File

The output file consist of a number of different components. In the (x, y) space it will have the same dimensions as the input cube. In the z direction it will have one pixel for each of the maximum number of parameters allowed to be fit (currently this is 4) and an extra 4 pixels which are flags used by TAU_FITS, and which may be of some use to you. The errors in the parameters are also stored in an error array. Note that the error array also covers the flags, but, of course, these don't have errors! For each (x, y) pixel the z pixels go as follows:

1. **Peak** – The maximum intensity of the fit *above* the background.
2. **Position** – The position of the peak in Å
3. **Width** – The width of the function (e.g. σ of a Gaussian or Cauchy function) in Å
4. **Background** – This is the level of the continuum. It is always assumed to be constant.
5. **Monitor** – This is an internal flag which is extremely useful for TAU_FITS, but of no interest to the user.
6. **Fit type** – This is the fitting function. It can have the following values: (1) Gaussian, (2) Cauchy function.
7. **Fit flag** – This tells whether a successful fit has been obtained for the pixel. It can have the following values: (-3) Fit failed in NEIGHBOUR, (-2) Fit was ZAPped by hand (in EDIT), (-1) Fit was rejected automatically (either by ZAP in main menu or it was considered to be unphysical in NEIGHBOUR), (0) Fit wasn't even attempted, (1) Fit done successfully, (2) Fit was successfully done by hand (in EDIT).
8. **Boxsize** – This is the smoothing box size for that pixel.

A Detailed programme descriptions

This appendix is intended to give a description of all of the TAURUS data reduction programs available and the parameters required for them. Throughout this appendix parameters which have been marked with an asterisk (*) are parameters which will only be requested if the user is *not* using a TAUCAL file. Parameters which have been marked with a dagger (†) are hidden parameters which may be changed on the command line but which are not prompted for.

A.1 CALBASIC

Function : Set up basic TAUCAL file with information about the telescope, TAURUS and its set up.

Description : CALBASIC gives you a lists of telescope setups, etalons and detectors. You are asked to indicate which in the lists are appropriate for your data.

Parameters :

CALOUT	Fixed string	This is the name of the TAUCAL file to be created with this program.

A.2 CALRUN

Function : Sets up .RUN substructures in TAUCAL file to hold information for specific TAURUS files.

Description : CALRUN gives you the option of creating four different run types– OBS (cubes with the 'object'), CAL (cubes with the calibration source), MASK (frames with the pinhole mask) and RING (individual ring frames for RVT). Depending upon the run type, the user is then prompted for other information about the structure. Note that the structures must exist as CALRUN tries to get dimensional information for some of them.

Parameters :

CALOUT	Fixed string	This is the name of the output TAURUS calibration file as created in CALBASIC.
RUNTYPE	Fixed string	This is the type of run to be added. They can be: MASK, OBS, CAL, or RING.
NSTRUCT	Integer	This is the number of structures of the type you've just nominated to be added to the TAUCAL file.
RUN_NOS	Integer array	This is the array of run numbers which correspond to the new substructures which you wish to create.
AGAIN	Logical	Instruction to continue adding further structures.
IMAGE	Fixed string	This is the name of the image for a particular run number.

MPU0	Float	This is the Z axis zero point in micro-processor units
DMPU	Float	This is the Z axis increment in micro-processor units
WAVE	Float	This is the wavelength for a particular run

A.3 CORRECT

Function : Does the actual phase correction of a TAURUS cube.

Description : CORRECT takes the phase map as defined in PHASE_MAP and shifts each spectrum. The sense is such that spectra with a positive phase are shifted to *lower* pixel values (e.g. CORRECT subtracts in wavelength space). The user is given a choice of two interpolation schemes. The input cube *must* be ZXY sorted before using this program.

Parameters :

USE_TAU	Logical	This is a flag to indicate whether a TAUCAL file will be used in this application.
CALFILE	Fixed string	This is the name of the TAUCAL file
OBSRUN	Integer	This is the run number associated with the OBS cube
NAME_OBS	Fixed string	This is the observation cube file. The cube must have been ZXY sorted before using this program!
USE_GRID	Logical	This is a flag to indicate whether you want to use a predetermined grid of X,Y coordinates as calculated in MASK and GRID which will automatically take into account the detector distortions.
NAME_GRID*	Fixed string	The name of the output file from GRID.
NAME_COR	Fixed string	This is the output file for the corrected data. Note that it will also be ZXY sorted as the input file is.
DX*	Float	This is the offset in X for the obs cube as found in EXAM_RVT
DY*	Float	This is the offset in Y for the obs cube as found in EXAM_RVT
XCEN*	Float	This is the X position of the optical axis
YCEN*	Float	This is the Y position of the optical axis

DMPU*	Float	This is the increment on the spectral axis in MPUs/frame.
NAME_SHIFTS*	Fixed string	This the output file for the final shifts. The shifts are defined as the total number of pixels to shift the spectra down in order to get the desired wavelength into the centre of the cube. (e.g. the shifts are subtracted from the spectra).
FSR_OBS*	Float	This is the free spectral range in MPUs of the OBS cube
PIXSIZE_X*	Float	This is the pixel size in microns.
CAMERA_FOCUS*	Float	This is the focal length in microns of the camera used in the observations.
INTERP	Integer	This is the interpolation scheme to be employed. At present these are (1) linear and (2) sinc. For integer data a sinc can sometimes cause rounding problems which lead to a ring pattern's appearing in the background.
WHICH_MASK	Logical	If there are more than one MASK in TAUCAL, then this decides which one to use.
WHICH_GRID	Logical	If there are more than one GRID in TAUCAL, then this decides which one to use.

A.4 CRE_RV

Function : Create estimated RVF or RVT files from information in TAUCAL file.

Description : CRE_RV was designed to be used in the event that the user has somehow lost the file with the ring parameter estimates created in PRERING (e.g. deleted it, run RINGS with the wrong parameters, etc) and he/she/it doesn't want to have to rerun PRERING. The information can all be painlessly recovered from TAUCAL or can be entered by hand. The latter may be rather tedious!

Parameters :

USE_TAU	Logical	This is a flag to indicate whether a TAUCAL file will be used in this application.
CALFILE	Fixed string	This is the name of the TAUCAL file.
RVF	Logical	Flag. True if the output file is an RVF and false if it will be an RVT.
WHICH_RVT	Integer	Which RVT from TAUCAL to use this time?

WHICH_RVF	Integer	Which RVF from TAUCAL to use this time?
MPU0*	Float	This is the value in MPUs that the first Z pixel will have...
DMPU*	Float	Increment in MPUs/pixel...
NSER*	Integer	This is the number of series in the cube to be processed.
SER_NOS*	Integer array	These are the first and last frame numbers which one wants to consider in any given series.
NZ*	Integer	This is the number of frames in the CAL cube
NAME_RVF*	Fixed string	This is the output file for the RVF.
NAME_RVT*	Fixed string	This is the output file for the RVT.
RING_PARS*	Float array	(1) MPU or Frame no., (2) Xc, (3) Yc, (4) R, (5) SIG, (6) A (7) B, (8) Ellipt, (9) Error, (10) Flag
NRUNS*	Integer	This gives the number of rings which you want to consider for the RVT.
RUNS*	Integer array	These are the run numbers to be included for this RVT

A.5 EXAM_RVF

Function : Examines the RVF array and calculates the free spectral range, the nominal gap and the mean centre of the rings (e.g. the coordinates of the optical axis).

Description : EXAM_RVF is an interactive program hence the user must have access to some sort of display. The user is presented with a plot of R^2 vs. Z. Because the Airy function is basically a paraboloid of revolution this plot should look linear. The user then marks where he wants each series to begin and end. A line is fitted to the points and the user is given a chance to delete or reweight any of the interior points with the cursor. If only one series is available, then the user must provide an estimate of the FSR. If only one series is any good, then the user may constrain all of the slopes of the fitted lines to be equal to the slope of the good series. The free spectral range is calculated from the differences in the x-intercepts. The gap is calculated from the FSR, the wavelength and the spacial scale. The coordinates of the centre are calculated from a weighted mean of all of the rings which have been used.

Parameters :

USE_TAU	Logical	This is a flag to indicate whether a TAUCAL file will be used in this application.
CALFILE	Fixed string	This is the name of the TAUCAL file

NSER*	Integer	This is the number of series in the cube to fit
WHICH_RVF	Logical	If there are more than one RVF in TAUCAL, then this decides which one to use.
WHICH_MASK	Logical	If there are more than one MASK in TAUCAL, then this decides which one to use.
NAME_RVF*	Fixed string	This is the input RVF file as determined in RINGS.
PIXSIZE_X*	Float	This is the pixel size in microns along the x direction in the middle of the frame.
CAMERA_FOCUS*	Float	This is the focal length in microns of the camera used in the observations.
WAVE_CAL*	Float	This is the wavelength of the source used for the calibration rings.
SAME_SLOPE	Logical	If there isn't enough information in one of the series to fit a reliable line, then you can constrain the subsequent lines to have the same slope as the first one.

A.6 EXAM_RVT

Function : Finds the offsets in Z, X_c and Y_c by an analysis of the RVT.

Description : EXAM_RVT uses the slopes found in EXAM_RVF to calculate a correction in Z resulting from the variation in R over time (δT being the time between the taking of the CAL cube and the taking of the OBS cube). Quite often 'time' is not the important factor in whether R has changed, rather it is whether the telescope has been moved. Hence the user is given a choice of 4 different ways to estimate the shift δR. These are described in section 4.6.

Parameters :

USE_TAU	Logical	This is a flag to indicate whether a TAUCAL file will be used in this application.
CALFILE	Fixed string	This is the name of the TAUCAL file.
OBSRUN	Integer	This is the run number associated with the OBS cube
WHICH_RVT	Integer	This is the number of the RVT to use this time.
WHICH_RVF	Integer	This is the number of the RVF to use this time.

EST	Integer	This is the method of estimation to be used. There are four possibilities (1) use the estimate from the ring just before the OBS run, (2) use the estimate from the ring just after the OBS run, (3) use an average of the above, (4) do an interpolation of the above. If you are doing this for the cal cube, then you should use method (1).
DISP	Logical	You may, if you wish, display R, X_c and Y_c versus time.
IGNORE	Logical	You may, if you wish, ignore the variation of R (and its implied variation in Z) and just do this calculation for Xc and Yc.
NAME_RVT*	Fixed string	This is the name of the RVT file as opened in RINGS.
XCEN*	Float	This is the X position of the optical axis.
YCEN*	Float	This is the Y position of the optical axis.
NSERFIT*	Integer	This is the total number of series actually fit in EXAM_RVF
FITDATA*	Float array	This is the series number, slope, its error, intercept and its error from each series. Actually all that is needed is the slopes and their errors so you can get away with typing rubbish for the other four...

A.7 FINDPKS

Function : Finds peaks in a pinhole mask frame, centroids them and writes out the positions.

Description : FINDPKS first does some rough statistics on the pinhole frame to find a noise level. It then locates pixels which are a user-supplied threshold number of σ above the mean background value. It does a test to make sure these are not just noise spikes. It then centroids the peaks it has found. The user has the option (and this is a highly recommended option!) to get a grey plot of the matrix frame with the located peaks indicated. He/she/it may then delete any peaks which may seem spurious (ghost images and the like). The final results are written to a file called FINDPKS.OUT and (if it's being used) to the TAUCAL file.

Parameters :

USE_TAU	Logical	This is a flag to indicate whether a TAUCAL file will be used in this application.
CALFILE	Fixed string	This is the name of your TAUCAL file for this data.

NAME_MAT*	Fixed string	This is the file with the matrix mask image (e.g. pinhole mask). The file must be 2-d.
MASKRUN	Integer	This is the run number you have assigned the matrix image in the TAUCAL file.
DISP	Logical	If you want to display the image, you will get a grey scale plot of the mask. The peaks which the program has found will overlay the image with open circles. You will then be given the chance to add or delete any points as you wish. Those which you have deleted will overlay as solid circles.
NRAD	Integer	This is the radius from the pixel of maximum intensity out to which you want to calculate the centroid of the peak.
THRESH	Float	This is the number of sigma above the clipped background used for defining which pixels can be considered as possible peaks.

A.8 GRID

Function : Generates a pixel grid to take account of flexure within the detector.

Description : GRID takes the polynomial fits from TMASK and generates a grid of new 'positions' for each pixel. The user can subset or recentre the grid. Extrapolating the fits is allowed, but not recommended.

Parameters :

USE_TAU	Logical	This is a flag to indicate whether a TAUCAL file will be used in this application.
CALFILE	Fixed string	This is the name of the TAUCAL file.
WHICH_MASK	Integer	This is the number of the mask to use if there are more than one in the TAUCAL file.
NAME_MAT*	Fixed string	This is the file with the matrix mask image (e.g. pinhole mask). The file must be 2-d.
NAME_GRID	Fixed string	This is the output file. Its dimensions will by (2,nx,ny) where nx,ny are the dimensions of the input matrix image.
DIFFSIZE	Logical	True if the output file is smaller or larger than the input file
CENTRED	Logical	True if the output file is centred on the MATRIX file.

NX	Integer	X dimension of output file
NY	Integer	Y dimension of output file
XC	Float	X position of output file centre. This is relative to the coords of the matrix file.
YC	Float	Y position of output file centre. This is relative to the coords of the matrix file.
NORDER*	Integer	Pretty self-explanatory, really...As this is a bivariate polynomial, the order will be the same in both coordinates.

A.9 LAMDA

Function : Does the wavelength calibration of TAURUS cubes

Description : LAMDA does the wavelength calibration in the following way (where upper case letters refer to the OBS cube and lower case to the CAL cube): the FSR (f) in pixels, gap (ℓ) and central phase (ϕ_o) are known for the CAL cube from EXAM_RVF. The FSR can also be calculated by:

$$f' = \frac{\lambda^2}{2\ell} \tag{7}$$

where λ and ℓ are both in microns. Thus the dispersion in the CAL cube is:

$$\delta\lambda = \frac{f'}{f} \tag{8}$$

and

$$\lambda_o = \lambda - \phi_o \delta\lambda.$$

The Fabry-Perot order is defined as:

$$n = \frac{2\ell}{\lambda}.$$

Because of errors in the calculation of ℓ this will not in general be an integer, which, of course it should be. So we take:

$$n' = [n]$$

where [] refers to the nearest integer and define a new estimate of the gap:

$$\ell' = \frac{n'\lambda}{2}.$$

Therefore,

$$N = \mathrm{int}\left(\frac{2\ell'}{\Lambda}\right)$$

and

$$\Lambda_o = \frac{n'\lambda_o}{N}.$$

The FSR of the OBS cube can be defined in pixels by:

$$F = \frac{\Lambda f}{\lambda}$$

and in microns (F') in the same way as equation 7 using Λ and ℓ'. Thus the dispersion in the OBS cube $\Delta\Lambda$ can be found ala equation 8. The wavelength zeropoint in the eventual *corrected* cube is found by assuming that you want Λ to be centered in wavelength space. Finally, the central phase in the OBS cube can be found by:

$$\Phi_o = \frac{\Lambda - \Lambda_o}{\Delta\Lambda} \quad (9)$$

All of these results are printed to the terminal.

Parameters :

USE_TAU	Logical	This is a flag to indicate whether a TAUCAL file will be used in this application.
CALFILE	Fixed string	This is the name of the TAUCAL file
NOBS	Integer	This is the number of OBS cubes to be calibrated.
OBSRUNS	Integer array	This is the run numbers for all the obs cubes to be calibrated
REPORT†	Logical	This is a flag to indicate whether a report will be given for all the cubes or just the first one.
WAVE_CAL*	Float	This is the wavelength of the source used for the calibration rings.
FSR_CEN*	Float	This is the free spectral range in MPUs of the CAL cube
GAP*	Float	This is the nominal gap in microns.
CENPHASE*	Float	This is the central phase of the CAL cube.
MPU0_CAL*	Float	This is the zeropoint on the spectral axis in MPUs in CAL cube.
DMPU_CAL*	Float	This is the increment on the spectral axis in MPUs/frame in CAL cube.
MODE*	Integer	This should be 1, unless the slopes from EXAM_RVF are negative. In that case it should be -1.
NZ*	Integer	Number of pixels in the spectral direction.
WAVE_WANT*	Float	This is the wavelength which you will eventually want to see in the centre of the calibrated OBS cube.
MPU0_OBS*	Float	This is the zeropoint on the spectral axis in MPUs in OBS cube.

DMPU_OBS*	Float	This is the increment on the spectral axis in MPUs/frame in OBS cube.
WHICH_RVF	Logical	If there are more than one RVF in TAUCAL, then this decides which one to use.

A.10 PEAKDISP

Function : Displays the MASK frame and the peaks found on it.

Description : PEAKDISP displays the MASK frame and the peaks which were located on it. This is useful in the event that the user has spotted something peculiar and wants to see exactly which positions are being fit in MASK. This program obviously requires a 2-d image display.

Parameters :

USE_TAU	Logical	This is a flag to indicate whether a TAUCAL file will be used in this application.
CALFILE	Fixed string	This is the name of your TAUCAL file for this data. You must have a TAUCAL file for this program otherwise the following programs won't be able to find the peak information.
MASKRUN	Integer	This is the run number you have assigned the matrix image in the TAUCAL file.
WHICH_MASK	Integer	This is the number of the mask sub-structure to use if there are more than one in the TAUCAL file.
NAME_MAT*	Fixed string	This is the file with the matrix mask image (e.g. pinhole mask). The file must be 2-d.
THRESH*	Float	This is the number of sigma above the clipped back ground used for defining which pixels can be considered as possible peaks.

A.11 PHASE_MAP

Function : Defines the phase map for an individual cube

Description : The theoretical phase map is defined by:

$$\Phi(x, y) = \Phi_o + \frac{2F\ell}{\Lambda}(\sec\theta - 1) \quad (10)$$

where Φ_o is the central phase, F is the free spectral range and Λ is the wavelength of the OBS cube. ℓ is the etalon gap and θ is the angle defined from optical axis to the pixel in question. Thus for a given spectrum the total shift necessary to put Λ in the centre of the cube (in the wavelength direction) is:

$$\mathcal{S} = \Phi(x,y) + d\Phi(x,y) + dZ - \Gamma \tag{11}$$

where $d\Phi(x,y)$ is the residual phase for that pixel found by RESIDUAL, dZ is the RVT shift for that cube found by EXAM_RVT and Γ is defined as the mean position in the spectral direction. What all this means is that in CORRECT an output pixel $O(z,x,y)$ will map onto the input pixel $I(z+\mathcal{S},x,y)$ or, in other words, CORRECT takes a pixel and *subtracts* the shift $\mathcal{S}$ to obtain the new position in z.

Parameters :

USE_TAU	Logical	This is a flag to indicate whether a TAUCAL file will be used in this application.
CALFILE	Fixed string	This is the name of the TAUCAL file
OBSRUN	Integer	This is the run number associated with the OBS cube
USE_RES	Logical	This is a flag to indicate whether you want to use a residual phase map. If you don't then the map is assumed to be all zeros
NAME_RES	Fixed string	This is the name of the file with the residual phase map.
USE_GRID	Logical	This is a flag to indicate whether you want to use a predetermined grid of X,Y coordinates as calculated in MASK and GRID which will automatically take into account the detector distortions.
NAME_GRID*	Fixed string	The name of the output file from GRID.
NAME_OBS*	Fixed string	The name of the OBS cube file.
WAVE_WANT*	Float	This is the wavelength to be centred (eventually) in OBS cube
DX*	Float	This is the offset in X for the OBS cube as found in EXAM_RVT
DY*	Float	This is the offset in Y for the OBS cube as found in EXAM_RVT
DZ*	Float	This is the offset in Z for the OBS cube as found in EXAM_RVT
XCEN*	Float	This is the X position of the optical axis
YCEN*	Float	This is the Y position of the optical axis
CENPHASE_OBS*	Float	This is the central phase of the OBS cube.
MPU0*	Float	This is the spectral axis zero point in micro processor units

DMPU*	Float	This is the increment on the spectral axis in MPUs/frame.
FSR_OBS*	Float	This is the free spectral range in MPUs of the OBS cube
GAP*	Float	This is the nominal gap in microns.
MODE*	Integer	This should be 1, unless the slopes from EXAM_RVF are negative. In that case it should be -1.
PIXSIZE_X*	Float	This is the pixel size in microns.
CAMERA_FOCUS*	Float	This is the focal length in microns of the camera used in the observations.
NAME_SHIFTS	Fixed string	This the output file for the final shifts. The shifts are defined as the total number of pixels to shift the spectra down in order to get the desired wavelength into the centre of the cube. (e.g. the shifts are subtracted from the spectra).
WHICH_RVF	Logical	If there are more than one RVF in TAUCAL, then this decides which one to use.
WHICH_MASK	Logical	If there are more than one MASK in TAUCAL, then this decides which one to use.
WHICH_GRID	Logical	If there are more than one GRID in TAUCAL, then this decides which one to use.

A.12 PRERING

Function : Does an initial guess at ring parameters

Description : PRERING is an interactive program only and requires a 2-d image display. An exhaustive description of PRERING is given in section 4.4.

Parameters :

USE_TAU	Logical	This is a flag to indicate whether a TAUCAL file will be used in this application.
CALFILE	Fixed string	This is the name of the TAUCAL file.
RINGRUN	Integer	This is the run number for either the CAL or RING structure.
CALRUN	Integer	This is the run number for the CAL cube whose frame is to be used as the reference ring in the RVT

CALFRAME	Integer	This is the frame number in the CAL cube for the frame which is to be used as the reference ring in the RVT
NAME_CAL*	Fixed string	This is the input file with the ring/s to be fitted. It can be either 2-d or 3-d depending on whether you are doing the RVF or the RVT relation.
TABLE	Fixed string	This is the name of the LUT (Look-up Table) for the 2-d image display.
MPU0*	Float	This is the value in MPUs that the first Z pixel will have...
DMPU*	Float	Increment in MPUs/pixel...
SER_NOS	Integer array	These are the first and last frame numbers which one wants to consider in any given series.
FRAMES_PROC	Integer array	These are the frame numbers for the 2 rings in the current series will be processed.
NFRAME	Integer	This is the frame number of a cube to process in case you are doing an RVT with rings in a cube.
NBINS†	Integer	This is the number of bins used in histogramming the data frame in intensity.
NSER	Integer	This is the number of series in the cube to be processed.
NAME_RVF	Fixed string	This is the output file for the RVF.
NAME_RVT	Fixed string	This is the output file for the RVT.
RVF	Logical	Flag. True if the output file is an RVF and false if it will be an RVT.
USE_ALL	Logical	Flag to use all the available rings for this RVT
NRUNS*	Integer	This gives the number of rings which you want to consider for the RVT.
RUNNOS*	Integer array	These are the run numbers to be included for this RVT

A.13 PRINT_RV

Function : Print RVF or RVT to an ASCII file.

Description : PRINT_RV will make an ASCII version of either the RVF or RVT. The output file name will be the same as the input, except with the extension .OUT substituted.

Parameters :

NAME_RV	Fixed string	This is either an RVF or an RVT file.

A.14 RESIDUAL

Function : Calculate and output the residual phase map.

Description : RESIDUAL takes the theoretical phase for a spectrum in the CAL cube ala equation 10 and calculates the residual from a moments analysis of that spectrum. Results can be binned in the spacial direction.

Parameters :

USE_TAU	Logical	This is a flag to indicate whether a TAUCAL file will be used in this application.
CALFILE	Fixed string	This is the TAUCAL file name.
NAME_CAL*	Fixed string	This is the name of the file with the calibration cube to be used.
WINDOW	Integer	This is the area in Z over which a peak will be searched. It is centred upon the predicted position of the peak as defined by the theoretical phase map. Thus the total space in Z to be searched will be 2*window + 1 pixels.
SMOOTH_SIZE	Integer	This is a box over which the results are smoothed.
USE_GRID	Logical	This is a flag to indicate whether you want to use a predetermined grid of X,Y coordinates as calculated in MASK and GRID which will automatically take into account the detector distortions.
NAME_GRID*	Fixed string	The name of the output file from GRID.
NAME_RES	Fixed string	The name of the output file for the residuals map.
PIXSIZE_X*	Float	This is the pixel size in microns.
CAMERA_FOCUS*	Float	This is the focal length in microns of the camera used in the observations.
WAVE_CAL*	Float	This is the wavelength of the source used for the calibration rings.

FSR_CEN*	Float	This is the free spectral range of the CAL cube in MPUs.
GAP*	Float	This is the etalon gap in microns.
XCEN*	Float	This is the X position of the optical axis
YCEN*	Float	This is the Y position of the optical axis
CENPHASE*	Float	This is the central phase of the CAL cube
MPU0*	Float	This is the spectral axis zero point in micro processor units
DMPU*	Float	This is the spectral axis increment in micro processor units per frame
MODE*	Integer	This should be 1, unless the slopes from EXAM_RVF are negative. In that case it should be -1.
DX*	Float	The mean RVT shift in X_c for CAL cube.
DY*	Float	The mean RVT shift in Y_c for CAL cube.
WHICH_RVF	Logical	If there are more than one RVF in TAUCAL, then this decides which one to use.
WHICH_MASK	Logical	If there are more than one MASK in TAUCAL, then this decides which one to use.
WHICH_GRID	Logical	If there are more than one GRID in TAUCAL, then this decides which one to use.

A.15 RINGS

Function : Does proper fits to calibration rings.

Description : RINGS does a proper fit to the calibration rings. The fitting function is of the form:

$$I(x,y) = Ae^{-(\rho-r)^2/2\sigma^2} + B$$

where

$$\rho = \sqrt{(x-x_c)^2 + (y-y_c)^2\epsilon^2}$$

Thus there are seven unknowns: A and B (the peak intensity and the background); x_c and y_c (the coordinates of the centre of the ring); r (the radius of the ring); σ (the width of the ring); and ϵ (the axial ratio of the ring).

Parameters :

USE_TAU	Logical	This is a flag to indicate whether a TAUCAL file will be used in this application.

CALFILE	Fixed string	This is the name of the TAUCAL file.
RINGRUN	Integer	This is the run number of the structure containing the rings to be fit.
NAME_CAL*	Fixed string	This is the input file with the ring/s to be fitted. It can be either 2-d or 3-d depending on whether you are doing the RVF or the RVT relation.
NAME_RVF*	Fixed string	This is the output file for the RVF.
NAME_RVT*	Fixed string	This is the output file for the RVT.
RVF	Logical	Flag. True if the output file is an RVF and false if it will be an RVT.
COMP_FAC	Integer	This is a factor by which the input data will be compressed. If it = 1 then all the data is used. If it = 2 then only every second row and column is used, e.g. 1/4 of the data.
WEIGHT	Integer	Flag to determine which weighting scheme gets used. (1) $\sqrt{n}$ (2) Equal weights
TRIVIAL†	Float	This parameter is used to test for the convergence of the fit. The fit is judged by how much χ^2 has changed over the last fit. TRIVIAL is thus the definition of a trivial change in χ^2 .
NTRIV†	Integer	This is the number of trivial changes to χ^2 needed to define the fit as converged.
RESTRICT	Logical	Flag. True if you want to restrict the region of analysis to an annulus of total width 2SIGSσ centred on the ring. The σ used is that of the estimate from PRERING.
SIGS	Float	This is the number of σ to define the width of the annulus which will be used if the user wishes to restrict the analysis region.
USE_GRID	Logical	This is a flag to indicate whether you want to use a predetermined grid of X,Y coordinates as calculated in MASK and GRID which will automatically take into account the detector distortions.
NAME_GRID*	Fixed string	The name of the output file from GRID.
TOLCENT	Float	This is a user supplied tolerance for the error in the central pixel determination. If the error is over this tolerance, then the fit is flagged in the output file.

TOLRAD	Float	This is a user supplied tolerance for the error in the radius determination. If the error is over this tolerance, then the fit is flagged in the output file.
NRUNS*	Integer	This is the number of rings for the RVT you want to define.
NAME_USED*	Fixed string	This is the ring file used in PRERING to set the initial guesses for the ring parameters. Note that this is only used to get the array sizes.
WHICH_RVT	Logical	If there are more than one RVT in TAUCAL, then this decides which one to use.
WHICH_RVF	Logical	If there are more than one RVF in TAUCAL, then this decides which one to use.
WHICH_MASK	Logical	If there are more than one MASK in TAUCAL, then this decides which one to use.
WHICH_GRID	Logical	If there are more than one GRID in TAUCAL, then this decides which one to use.
WHICH_FRAME	Logical	If you are doing RVTs from cubes, then define which frame to analyse.
CIRCLE†	Logical	This is a flag to constrain the axial ratio of the rings to 1.0

A.16 SIZES

Function : Write a 2-d map of pixel sizes.

Description : SIZES calculates the size in either X or Y of a pixel as a function of its position and puts the results into a 2-d map which is suitable for display.

Parameters :

USE_TAU	Logical	This is a flag to indicate whether a TAUCAL file will be used in this application.
CALFILE	Fixed string	This is the name of the TAUCAL file.
AXIS	Integer	You have a choice of either mapping the X pixel size (axis = 1) or the Y pixel size (axis = 2).
NX	Integer	How big an output map do you want in X?
NY	Integer	How big an output map do you want in Y?
PIX_MAP	Fixed string	Name of the file to the map of pixel sizes.

NORDER*	Integer	Pretty self-explanatory, really...As this is a bivariate polynomial, the order will be the same in both coordinates.
WHICH_MASK	Integer	This is the number of the mask sub-structure to use if there are more than one in the TAUCAL file.

A.17 TAU_FITS

Function : Fit profiles to spectral lines.

Description : For a full description of TAU_FITS, see section 5.2

Parameters :

NAME_OBS	Fixed string	This is the phase corrected observation cube for input. Must be ZXY sorted BEFORE using this program.
NAME_FIT	Fixed string	This is the output file for the fit results. The fit parameters will be stored in a separate plane for each and the associated error array will give the parameter error. Magic values flag spectra which were either not analysed or failed to converge.
NITER	Integer	Maximum number of iterations for each spectrum.
OPT	Fixed string	This is the option parameter in the main menu.
BOXSIZE	Integer	Data will be averaged over a box of this size. For obvious reasons, boxsize should be odd, although it won't be fatal if it isn't
TYPE_FIT	Integer	This gives you the option of the fitting function for the spectra
REJ_DOM	Integer array	Type 1 to reject in a domain or 0 to leave that domain alone.
START	Float array	Lower limit on X and Y for action.
END	Float array	Upper limit on X and Y for action.
MASK_THRESH	Float array	If the mask has a value between the thresholds at a particular pixel position then that pixel will be analysed.
NAME_MASK	Fixed string	This is a 2-d mask image.

DLAM	Float	This is the wavelength increment along the spectral axis.
ALAM0	Float	This is the wavelength zeropoint along the spectral axis.
FSR	Float	This is the free spectral range of the obs cube at the optical axis
USE_MASK	Logical	If true, the you will be prompted for a mask image, lower and upper thresholds for masking when the limit routine is run.
NEW_FILE	Logical	If true, then the output file is a new file.
WT_SCHEME	Integer	The user may weight the points by (1) $\sqrt{n}$ or (2) equally.
HOW_GUESS	Integer	This is the choice of guessing algorithm to use. See section 5.2.2
NTRIV	Integer	This is the number of times that χ^2 must change by a trivial amount for the fit to be considered to have converged. This number should be greater than 1 as the first trivial change may be a false alarm.
DCHI	Float	This is a *fractional* change in chi-squared, e.g. $d\chi^2 = \left|(\chi_1^2 - \chi_2^2)\right|/\chi_1^2$. Therefore if $d\chi^2 = 0.05$, then there has been a 5 percent change in χ^2.
XTRACT_FIELD	Fixed string	This is the extracted field output file.
XTRACT_PAR	Integer	This is the parameter which you want to extract.
RNG_AMP	Float array	Range of amplitudes to keep.
RNG_POS	Float array	Range of peak positions to keep.
RNG_SIG	Float array	Range of sigma to keep.
RNG_BAC	Float array	Range of background to keep.
NSWEEP	Integer	This is the number of sweeps through the monitor array in order to try an get all of the included spectra...
BATCH_FIT	Integer	This is a flag which tells the program in batch what you ultimately want to do. (1) fit, using a guessing algorithm, (2) nearest neighbour analysis.
LAM_CEN	Float	This is the wavelength about which the velocity field is to be calculated

TOL	Float	This is the tolerance which defines whether or not two velocities are separated by an integer number of FSRs
VEL_FIELD	Fixed string	This is the file for the input/output velocity field
SCALE_Y	Logical	TRUE if you want to set the min and max values of the Y axis of all line plots to be the same (e.g. all line plots have the same scale)
YRANGE	Float array	This is the min and max values for the Y axes of the line plots.

A.18 TAU_HEAD

Function : Write calibration information into a cube header.

Description : TAU_HEAD takes information from a TAUCAL file and writes it into the header of a cube. If a TAUCAL file isn't being used, then the user will be prompted for this information (which may be rather boring). The results end up in the sub-structure .MORE.TAURUS and are very useful for TAU_FITS.

Parameters :

USE_TAU	Logical	This is a flag to indicate whether a TAUCAL file will be used in this application.
CALFILE	Fixed string	This is the name of the TAUCAL file
IMAGE	Fixed string	This is the file whose header you want to write
RUNNUM	Integer	This is the run number associated with the OBS cube
DLAM*	Float	This is the increment along the spectral axis.
ALAM0*	Float	This is the zeropoint along the spectral axis.
FSR*	Float	This is the free spectral range of the OBS cube at the optical axis
DMPU*	Float	This is the increment on the spectral axis in MPUs/frame.
XCEN*	Float	This is the X position of the optical axis
YCEN*	Float	This is the Y position of the optical axis
WHICH_RVF	Logical	If there are more than one RVF in TAUCAL, then this decides which one to use.

WHICH_MASK	Logical	If there are more than one MASK in TAUCAL, then this decides which one to use.
CAMERA_FOCUS*	Float	This is the focal length in microns of the camera used in the observations.
COL_FOCUS*	Float	This is the focal length in microns of the collimator of the telescope.
SCALE*	Float	This is the scale of the telescope in arcsec/mm
TEL_NAME*	Fixed string	This is the name of the telescope.
CAM_FNUM*	Float	This is the *f* number of the camera used for the observations.
TAU_VER*	Integer	This is the TAURUS version number.
PIXSIZE_X*	Float	This is the pixel size in microns.
GAP*	Float	This is the etalon gap in microns.

A.19 TMASK

Function : Find transformation needed to create a grid of uniform scale.

Description : TMASK takes the peaks found in FINDPKS and fits a bivariate polynomial to find the transformation between the positions of the peaks in pixel space and their positions on a grid of uniform scale. The latter has been defined by the known distance in microns between the pinholes in the mask. An output file called MASK.OUT contains the transformation coefficients and grids of the pixel sizes in each dimension and the axial ratio of the pixels as a function of position on the map. The coefficients are also stored in the TAUCAL file, if one is being used.

Parameters :

USE_TAU	Logical	This is a flag to indicate whether a TAUCAL file will be used in this application.
CALFILE	Fixed string	This is the name of your TAUCAL file.
WHICH_MASK	Integer	This is the number of the mask sub-structure you wish to use, if there are more than one in the TAUCAL file.
NAME_MAT*	Fixed string	This is the file with the matrix mask image (e.g. pinhole mask). The file must be 2-d.
MASK_DIST*	Float	This is the actual distance between the pinholes as projected onto the focal plane.

NORDER	Integer	Pretty self-explanatory, really...As this is a bivariate polynomial, the order will be the same in both coordinates.
CAMFOC*	Float	This is the focal length of the camera in mm.
COLFOC*	Float	This is the focal length of the collimator in mm.

B Main Menu Options in TAU_FITS

This is a complete listing of all of the available options for the main menu in TAU_FITS. Note that all options can be abbreviated to a single character.

Table 3: Main Menu Options in TAU_FITS

Parameter	*Description*
Boxsize	Change the size of the smoothing box
Converge	Change the convergence parameters
Edit	Edit the fits in the current (x, y) domain.
Fit	Do fits in the current domains
Guess	Change the guessing algorithm to be used
Help	Print up this help menu
Info	Print information about current set up.
Limits	Define the limits in the current domains
Mask	Decide whether or not to use a mask when setting limits
Neighbour	Do fits with nearest neighbour algorithm
Picture	Plot up a 2-d picture
Quit	Get out of TAU_FITS.
Show	Show the results for a single spectrum
Type	Change the type of fitting function
Weights	Change the way individual pixels are weighted
Velocity	Remove integral FSR jumps from velocity map
Xtract	Extract a set of results and write them to a file
Yscale	Make Y axis scale the same for all plots
Zap	Reject bulk fits

C Edit Menu Options in TAU_FITS

This is a complete listing of all of the available options for the EDIT menu in TAU_FITS. Note that all options can be abbreviated to a single character.

Table 4: EDIT Menu Options in TAU_FITS

Parameter	*Description*
Converge	Change convergence parameters
Fit	Do fit of current spectrum
Guess	Change the guessing algorithm to be used
Help	Print up this help menu
Info	Print up information on set up of program
Next	Move on to the next panel of spectra
Quit	Get out of EDIT mode
Previous	Look at the previous panel again
Redraw	Redraw the current panel
Show	Display the fit parameters for indicated spectrum
Type	Change the type of fitting function
Wipe	Delete all the fits on the current panel
Zap	Delete the current fit