Reduction Process in Sci-Phot Mode

0. Setup/Inputs
=============

The software **sptest.pro** is written by Walter Jaffe and uses the MIA+EWS package (written by Rainer Koehler and Walter Jaffe). The following description is partly based on the official web-manual, the MIDI wiki and private communications.

0.1. Data

For every observations in Sci-Phot mode one needs the photometry (taken directly before or after fringe tracking) and the interferometry files from the target and the calibrator. The dispersive element can be either PRISM or GRISM. Since MIDI lose already light observing in Sci-Phot mode (light is splitted between photometric and interferometric channels) and the calibrators are mostly relative faint, it is advisable to observe in PRISM mode. Also the described software need bright sources to work properly.

Mode:	**Sci-Phot**
Dispersive element:	**PRISM** or **GRISM**
Source brightness:	**UTs > 15 Jy, ATs > 200 Jy** (for target and calibrator)
Tracking mode:	Group Delay Tracking
Chooping:	Yes (photometry and interferometry observation; chooping frequence: 0.306 Hz [grism], 0.545 Hz [prism])

0.2. Fits files and filenames

For the next chapters the names have the following meaning:

IDL string variables which contain the list of fits file filenames:
photcalfile = AOPEN and BOPEN photometry files from the target or the calibrator
tarfile = ABOPEN interferometry files from the target
calfile = ABOPEN interferometry files from the calibrator (to calibrate visibility and flux)

One can access the files using the following command: file = midigui(dir=current_dir)

IDL string variables which just contain the real name of the source:
photcal = name of photometric calibrator (same as target or calibrator)
target = name of target (e.g. target = “VHya”)
calibrator = name of calibrator (e.g. calibrator = “HD19324”)

A = telescope one
B = telescope two
AB = light from both telescopes together

There are 4 detector arrays (each 171 by 40 pixel):

PA	photometric channel for telescope A
I1	interferometric channel 1
I2	interferometric channel 2
PB	photometric channel for telescope B

In the horizontal direction the light is dispersed (= λ-direction) and the vertical direction is the y-direction.

0.3. Used Routines

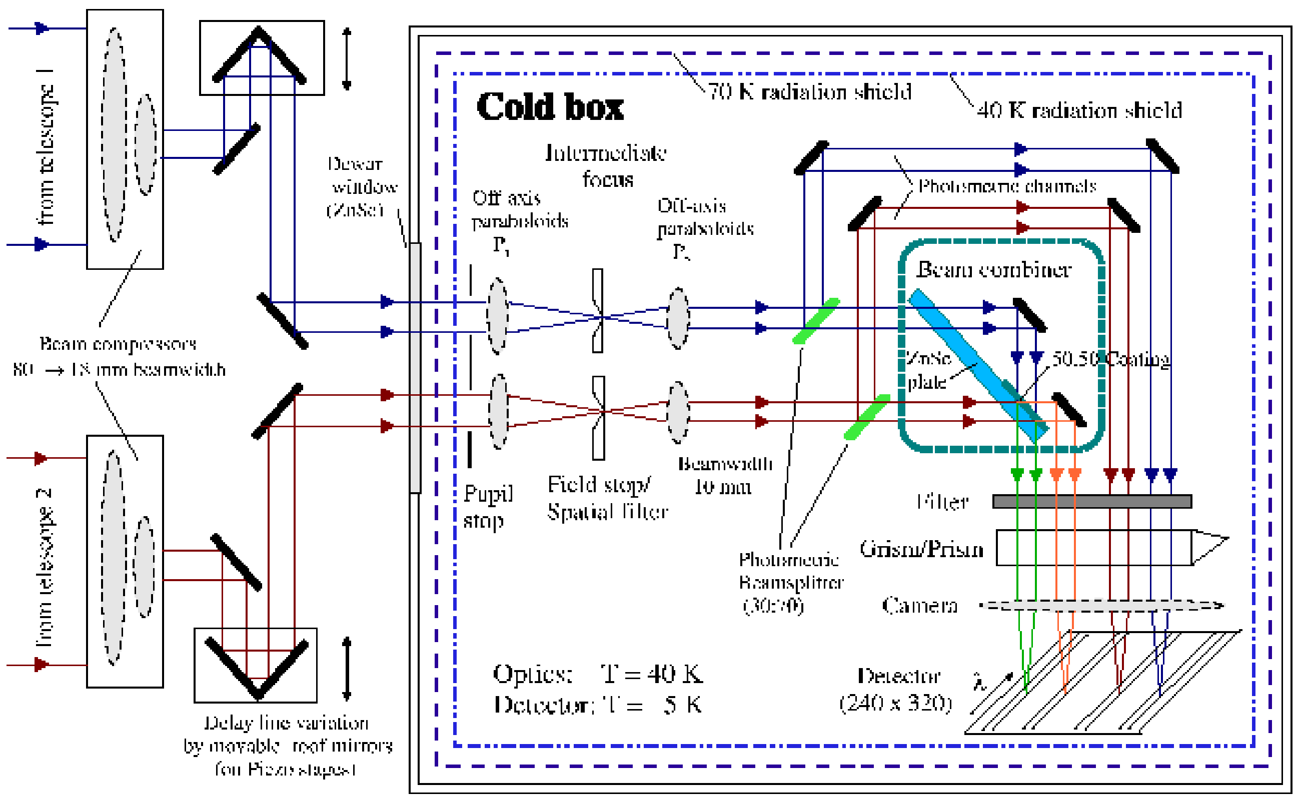
There are basically only four commands one have to execute. So a typical reduction would look like this:

> \> **spCrossCal, photcal, photcalfile**
> \> **spPipe, target, tarfile, photcal**
> \> **spPipe, calibrator, calfile, photcal**
> \> **spCalVis, target, calibrator**

In the following chapters these routines are explained but also the subroutines are mentioned. The fits files are normally accessible with data = oirGetData(filename).

0.4. MIDI Instrument

Here an overview of the MIDI instrument. Important components here are the photometric beam splitter, the beam combiner, the prism or grism and the detector array.



1. Calculation of the Cross Coupling Coefficients / Point Spread Function

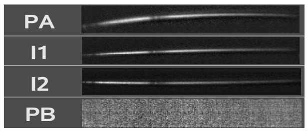
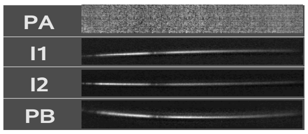
==

Routine: **spCrossCal, photcal, photcalfile**, smx=smx, fudge=fudge

First we have to calculate how much of the incoming light goes in the photometric and how much in the interferometric channel (= the split ratio between both = kappa coefficients). For this reason photometric observations were performed besides the interferometric observations. In this part of the observation always only one of the two incoming beams go into MIDI. In contrast to the High-Sense mode a photometric beamsplitter is inserted before the beam combiner. If now the light from telescope one (AOPEN) comes in, one part goes to the first photometric channel (PA) and the other part to the interferometric channel. Since there is still the beamsplitter in this channel, both interferometric channels are illuminated (I1 and I2). For telescope two open (BOPEN) PB, I1 and I2 are illuminated. With the following steps the 4 split ratios over the wavelength are calculated (I1/PA, I2/PA, I1/PB and I2/PB):

1.1. Unchop Data

In this step the AOPEN and BOPEN data for a strong photo-calibrator are (separately) unchopped using the routine **oirChopPhotoImages**. One can use either the photometric observations of the target or the calibrator – the brighter the better. With this step one gets rid of most of the overwhelming background () that dominates observations in the mid-infrared by subtracting the on source position by the off source position. This is done by observing with the telescope a nearby part of the sky: "chopping". The produced AOPEN and BOPEN source spectra look like this (average of all good input frames):

	***photcal.Achop.fits***	unchopped images from AOPEN data *
	***photcal.Bchop.fits***	unchopped images from BOPEN data *

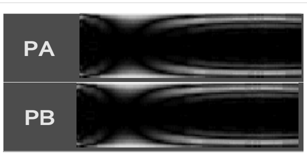
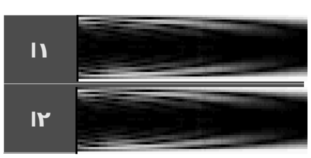
** [7 element array of structures, where each element inhabits the 4 channels as DATA1 to DATA4 and in the first element are the unchopped images (= source – sky, this is shown here) and in the last element are the sky images. The elements between are split-ups of the photometry of the first element (?)]*

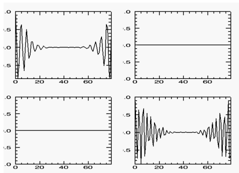
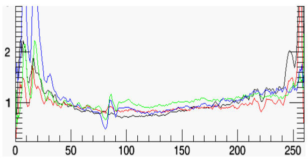
1.2. Estimate an Instrumental Spectrum in each Channel

Now the Point Spread Functions (PSF) for each channel in A and B open data are fitted with the routine **spCross** which can produce some error massages. They describe the y-profile as a function of wavelength. From this, the photometric masks and kappa coefficients are determined.

In this step (spCross) and in some later steps (Sci2Hi) a critical point is the determination and removel of the remaining background not subtracted by the unchopping (spFindTunnel). It is neccessary to find a region near the star spectra where only the sky background dominates. Therefor the routine have to destingish between tunnel, sky and source flux, and have to know the slit position. All these things are changing with different AT positions/baselines and pointing. This causes at the end problems in the callibration process since a too high or too low photomtric or interferomtric signal can lead to visibilities higher then 1.

The output files of spCross are:

	***photcal.psfab.fits***	y-profile in the PA and PB channel (normalized fourier transforms of photo-calibrator profiles in PA and PB channel of the A and Bchop observation) *[1 element array of structures with DATA1-4]*
	***photcal.psfs2.fits***	y-profile in I1 and I2 channel for **each** A and Bchop observation (normalized Fourier transforms of photo-calibrator profiles in I1 and 2 channel of each A and Bchop observation) *[2 element array of structures with DATA1-4]*

 (3x)	***photcal.psfsy.fits***	y-profile, averaged over several channels, in center of wavelength range (?) *[3 element array of structures with DATA1-4, each data with 80 elements]*
 (2x)	***photcal.mask.fits***	description of where to put masks in each of the 4 channels, from A and Bchop files (?) *[2 element array of structures with DATA1-4, each DATA 171x2 pixel]*
	***photcal.kappa.fits***	**estimates of I1/PA, I2/PA, I1/PB, I2/PB masked flux ratios based on the photometry***

* the photometric beamsplitter have a split ratio of about 30:70 and since the 70% are again splitted by the beam combiner the flux ratios are about 1.

2. Calculation of the Instrumental Visibility (Target and Calibrator)

==

Routine: **spPipe, target, tarfile, photcal**, curve=curve, smooth=smooth, $
gsmooth=gsmooth, abmask=abmask,nodelete=nodelete and

spPipe, calibrator, calfile, photcal, curve=curve, smooth=smooth, $
gsmooth=gsmooth, abmask=abmask,nodelete=nodelete

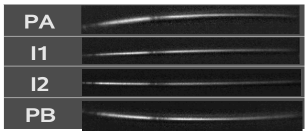
This operates now on the ABOPEN chopped fringe tracking data. This means now that the light from both telescopes are going into MIDI and each beam is splitted up with the photometric beamsplitter. One of each splitted beam produces the PA and PB photometry. The other one of each beam goes trough the beam combiner and produces the interference pattern (fringes) in the channel I1 and I2 (at the same time). Both interferometric channels are shifted by 180 degrees.

After some photometric reduction (unchopping, and applying the kappa coefficients) to the photometric data the interferometric data are reduced. This is done for the target **and** the calibrator separately (here it is shown in parallel).

2.1. Photometry

2.1.1. Unchop Images

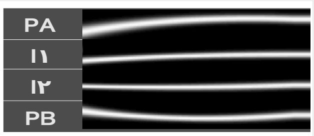
First, it unchops the ABOPEN target/calibrator data (interferometric and photometric channel) with **oirChopPhotoImages** to get again rid of most of the background. The target/calibrator unchopped images look than as follows:

	***target.ABchop.fits*** ***calibrator.ABchop.fits***	images of the dispersed fringes I1 and I2, and the simultaneous photometry PA and PB *

** [7 element array of structures, where each element inhabits the 4 channels as DATA1 to DATA4 and in the first element are the unchopped images (source-sky) and in the last element are the sky images.]*

2.1.2. Generate A Mask

Next, it compares the y-positions in the PA and PB channels of the target/calibrator versus the images in these channels for the previously processed phot-calibrator, and shift the mask positions for the photo-calibrator to match the target/calibrator pointing. It is done with **spMakeTrackMask**. The new constructed masks based on the phot-calibrator and ABchop files are:

	***target.ABmask.fits*** ***calibrator.ABmask.fits***	description of where to put masks in each of the 4 channels, from ABchop files *[1 element array of structures with DATA1-4]*

2.1.3. Calculate the Incoherent Flux ("Photometry")

Now one important step: calculate the I1 and I2 "photometry" images based on the flux in the PA and PB channels (project the PA and PB channel separately to the I1 and I2 channel and average). This uses the IDL routine **sci2Hi** (which converts basically the SCI-PHOT Photometry images into HIGH-SENSE photometry images) and is quite complicated:

A: Estimate from the PA and PB channels the shift in actual pointing between the target/calibrator and the photo-calibrator.

B: From this shift and the photo-calibrator PSF calculate "sky background" regions in the PA and PB channels and removes the remained sky background.

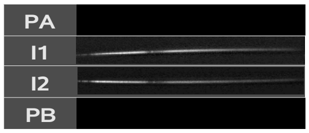
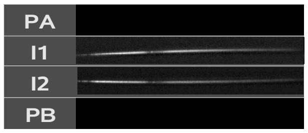
C: Fit wavelength-channel by channel in PA and PB the calibrator PSF in this channel to the measured target/calibrator signal in order to estimate the photometric flux.

D: Transfer this flux to the I1 and I2 channels:

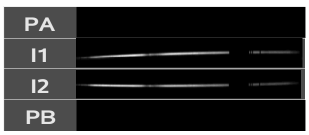
- i) Multiply PA and PB by the appropriate kappa coefficient to get I1_A, I2_A and I1_B, I2_B (where I1_A is the image predicted in I1 from the data in PA)

- ii) Shift in wavelength to correct for differences in PA/I1, PA/I2, and PB/I1, PB/I2 wavelength scale.

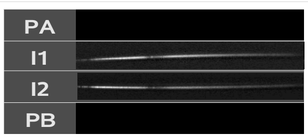
iii) Multiply by the I1 and I2 PSF from the photo-calibrator. Note that this produces a much sharper image in I1 or I2 than the old procedure of correcting PA and PB images for the y-curvature. That procedure is a source of error, since calculating the correct overlap is difficult. The calculated flux in I1 and I2 from PA and PB are now:

	***target.Aphotometry.fits*** ***calibrator.Aphotometry.fits***	estimated I1 and I2 image based on PA signal *[7 element array of structures with DATA1-4]*
	***target.Bphotometry.fits*** ***calibrator.Bphotometry.fits***	estimated I1 and I2 image based on PB signal *[7 element array of structures with DATA1-4]*

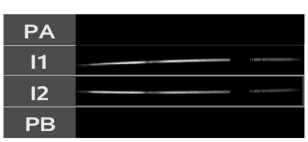
iv) Calculate the **geometric mean images SQRT(I1_A*I1_B) and SQRT(I2_A*I2_B)**:

	***target.geo.fits*** ***calibrator.geo.fits***	estimated SQRT(PA*PB) in I1 and I2 = uncorrelated flux in I1 and I2 *[7 element array of structures with DATA1-4, 6 used]*

v) Also calculate predicted **I1_A + I1_B and I2_B + I2_A** (correct?) image, which should approximate look like the data in *target/calibrator.ABchop.fits* (?). This is useful for estimating the instrumental photometric flux later.

	***target.ABphotometry.fits*** ***calibrator.ABphotometry.fits***	estimated I1 and I2 image as sum of PA+PB *[7 element array of structures with DATA1-4]*

vi) Optionally: correct for lost fine spectral structure data. The poor optics of the PA and PB channel results in poor spectral resolution in the corners of the image. Essentially it measures the photometry in the I1 and I2 channel where the optic is good by using the predicted A+B images from step (v). Then it corrects the SQRT(A*B) images so that the fine structure looks like the photometry in the I1 and I2 channels, but the smooth overall flux variation looks like that predicted from the PA and PB channel (?). The corrected geometric mean is then:

	***target.geo2.fits*** ***calibrator.geo2.fits***	estimated SQRT(PA*PB) in I1 and I2 = uncorrelated flux in I1 and I2 (corrected for loss of spectral resolution) *[1 element array of structures with DATA1-4]*

Now we have estimated the **uncorrelated flux** in the I1 and I2 channel (gotten with the kappa coefficients from the PA and PB channel). Everything has been photometric so far. Now we use the interferometry data to obtain the **correlated flux** for the calculation of the visibility.

2.2. Interferometry: Delay and Phase

In this part the *target/calibrator.ABmask.fits* file is used as mask to run the usual interferometric routine **midiVisPipe** to estimate instrumental, group and atmospheric delay, and phase behavior. The simple correlated fluxes, one also gets here, are not used later on. Only the estimated group delay is important for step 2.3.

2.2.1. Compression

The first reduction step is compression using **oir1dCompressData**. It reduces the 3-D data cube basically in a 2-D image, where one dimension represents the time and the other the wavelength (Each two dimensional detector image is summed up perpendicular to the PRISM/GRISM dispersion and the resulting one dimensional spectra (= frames) are stacked together in the time direction). The unchopping was already done some steps before.

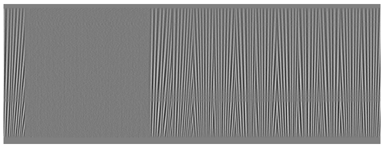
The output file *target/calibrator.compressed.fits* is deleted (cleaned up) at the end if the keyword “no delete” is not set.

2.2.2. Formation of Fringes and High-Pass Filter

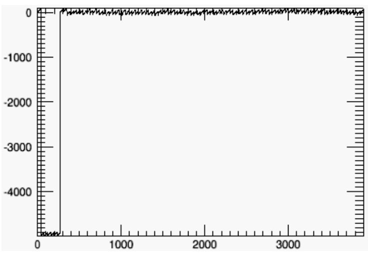
In the second step, using **oirFormFringes**, the two interferometric channels are subtracted (they are 180 degrees out of phase), which reduces the background by about 90%, while the spectrally dispersed fringes remain. It assumes that the wavelength channels are sufficiently aligned. At this point it also applies a high-pass filter to reduce sky and instrumental backgrounds that vary more slowly than the fringe (The fringe should vary quickly because of the OPD modulation): For each pixel in the compressed data, it runs a **boxcar** filter of given width in frame numbers over the data, and then subtract this smoothed version from the original version.

(The default boxcar width is 50 frames, but this can be overwritten by the **smooth** parameter in the call. If you use a very small smooth parameter [e.g. 2 or 3] some of the real interferometric signal will be removed by the filter, so the sensitivity goes down. But if the source is very weak and one discovers that the group delay finding step [below] finds low frequency garbage left over from the sky, then one should use a smaller value of smooth to suppress this. Note: be sure to use the same value of smooth for the target and for the calibrator!. Default value for smooth is 8.)

The output of this step is a simple image data structure: The vertical direction corresponds to the wavelength (fringes are dispersed) and the horizontal to the time. Since MIDI changes the OPD with the internal piezos with time (it scans through the fringe) this dimension corresponds also to the fringe position. This scans are repeated after every 40 frames (where one frame is one integration with a DIT of typically 18 ms [Prism] or 36 ms [Grism], and with a DITDELAY of 5 ms, so one scan takes about 1 or 2 seconds). At the begin of an observation the OPD is offset by a large value (one sees no fringes) in order to measure the noise (for what?).

	***target.fringes.fits*** ***calibrator.fringes.fits***	fringes per wavelength (only a part is shown here: the first 600 frames) *[3900 element array of structures with DATA1 (171 elements)]*

It is sometimes desired to read out the tracking OPD in microns, after combining all the possible delay lines. This can be done with opd = oirgetopd(filename). This works with almost all the data files during the reduction, since the OPD is copied from one to the next, but it is faster with data that are already compressed.

		tracking OPD *(large offset at the beginning, then the zig-zag pattern of the fringe scans)*

2.2.3. Remove instrumental OPD (Fourier Transform)

Next, the routine **oirRotateInsOpd** removes the known instrumental OPD by multiplying each data point at frequency k (in spatial units: 2 pi/lambda) of the previously produced fringe spectra by **exp(-ikOPD)**, where OPD is the sum of the OPD from the MIDI piezos and the VLTI delay lines. The output is again an imagedata structure, but the actual data are now complex, but represented by two float numbers. (To convert this IDL structure to a real IDL complex value and to transpose the array, so that time is in the x-direction and wavelength in the y-direction, which is usually nicer for display, use pseudoComplex. The absolute value can then be displayed)

The output file *target/calibrator.insopd.fits* is deleted (cleaned up) at the end if the keyword “no delete” is not set.

2.2.4. Group Delay Analysis

Now, **oirGroupDelay** searches for the group delay. The group delay is the Fourier Transform of the spectrum of the previous step (thus in delay domain, not in frequency domain). After Fourier Transform it averages several frames together to suppress one image peak and increases the S/N. It also looks for the position of the peak in the absolute value and writes this down for later use in step 2.3. A more detailed description follows now:

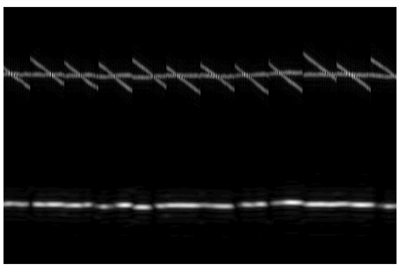
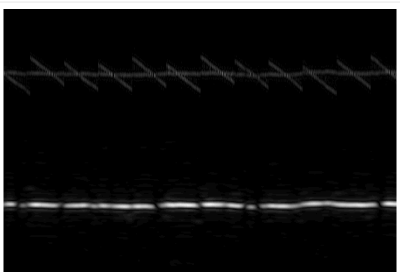
Note that because the MIDI beam-combiner has only two output phases it is essentially a cosine correlator and not a complex correlator. This means that the group delay has two peaks: one at positive and one at negative delays.

Because the actual OPD was modulated by the MIDI piezos and this modulation was de-rotated in oirRotateInsOpd before, one of the two peaks is nearly stationary from frame to frame (only atmospheric OPD movement remains) while the other peak moves around with twice the instrumental modulation. Thus if we average a few frames together the wrong side band is strongly suppressed.

The averaging is done in framenumber space, specified by the **gsmooth** (Gaussian smoothing) value and does not know anything about the actual time of each frame. It picks the standard deviation of a Gaussian in frame number that is run over the data to gsmooth it (basically frames within the coherence time are smoothed together). The default for gsmooth is **4** frames.

The best choice of gsmooth depends on the source and the weather. Strong sources in rapidly varying OPD weather should use a small value of gsmooth, but probably not smaller than 2 frames. Alternatively weak sources in good weather should use a larger value. For weak sources in bad weather, this is out of luck.

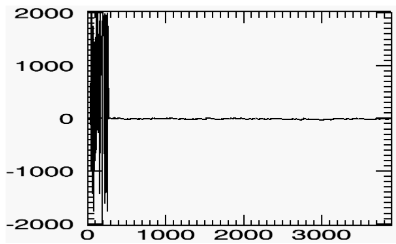
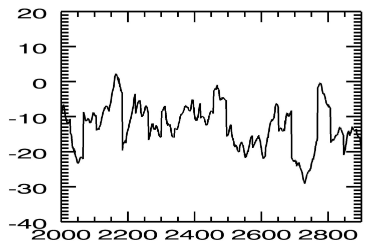
The result main data table is again pseudocomplex. The absolute value of the original and gsmoothed delay function as a function of delay (pixel number, in vertical direction) for every frame (horizontal direction) looks like this:

	***target.groupdelay.fits*** (original image and gsmoothed one)	delay function for the target *[3900 element array of structures with DATA1 (1024 elements)]*
	***calibrator.groupdelay.fits*** (original image and gsmoothed one)	delay function for the calibrator *[3900 element array of structures with DATA1 (1024 elements)]*

At the upper plot of each image there are two lines visible: a more or less straight one that corresponds to the correct instrumental OPD (the peak of the two which is stationary from frame to frame after de-rotation) and a saw tooth one (the peak of the two which now moves twice as fast as the instrumental delay). For faint objects there would be a second saw tooth visible with an angle between the straight line and the first saw tooth. This would correspond to an uncorrelated sky signal (and moves as fast as the instrumental delay after de-rotating).

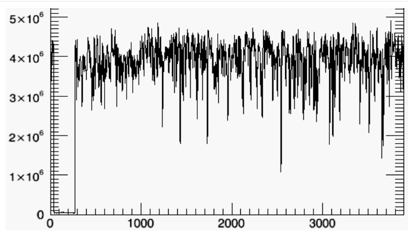
The lower gsmoothed plots show now the delay function with the removed wrong OPD (averaged frames with suppressed image peak and increased S/N). If one do not see a smooth, well behaved peak (in vertical direction), the data are probably worthless!

This files contain another interesting table: the delay table which shows the delay the fringe tracker has found in the above images (basically the brightest spot = position of the peak in the absolute value) for each frame.

		group delay (unzoomed and zoomed) *[with delay in µm]*

Here one can see how good the conditions were and therefore how good the S/N is. If there is not such a smooth line, e.g. one have big jumps and a lot of spikes, many frames are bad.

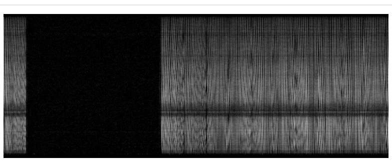
This file inherits also an amplitude table which looks like this:

		(fringe) amplitude

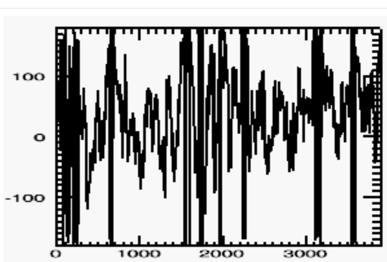
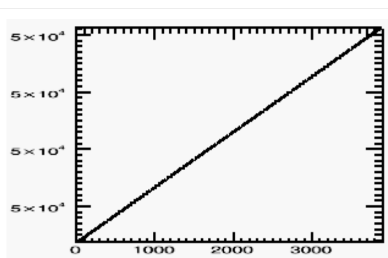
Now the group delay is determined and so the next steps are additional but done in a similar way later in part 2.3.! One can go directly to chapter 2.3 but should know the next steps.

2.2.5. Form aligned Frames

After locating the group delay, the routine **oirRotateGroupDelay** simultaneously removes the **group delay** and the **instrumental OPD** (MIDI piezos + VLTI delay lines) from the original fringe data produced in 2.2.2. The algorithm is the same used in 2.2.3., only that the group delay is now also removed. The frames are then aligned (de-rotated) and the output is again pseudocomplex:

	***target.ungroupdelay.fits*** ***calibrator.ungroupdelay.fits***	fringes per wavelength (only a part is shown here: the first 600 frames) *[3900 element array of structures with DATA1 (1024 elements)]*

Additionally there is also a dispersion table which contains the value of the average offset phase subtracted from each row (the phase which comes primarily from water vapor dispersion), as well as an time table which maps frame numbers to time.

		dispersion *[delay in µm over frame number]* and time *[in ms over frame number]*

2.2.6. Editing

It flags with the routine **oirAutoFlag** individual frames that seem wrong. This is not done on the basis of the estimated fringe amplitude (since dropping low amplitude frames biases the result), but on two other criteria: The **distance between the instrumental OPD and the group delay**, and the **existence of OPD jumps in the group delay** estimate.

A large difference between the instrumental OPD and the group delay OPD indicates that the on-line MIDI system had not (yet) found the fringe. The maximum allowable difference depends on the spectral resolution of MIDI being used: spectral resolution divided by the OPD difference, measured in compatible units, should be less than one radian. In practice this means that ΔOPD should be less than 100 µm for the PRISM and about 500 µm for the GRISM. In particular, there are 50 or so frames at the beginning of each tracking run where one deliberately set the OPD way off, and these should be rejected from the estimation.

Secondly, if there is a big OPD jump from one frame to the next, caused e.g. by the UT auto focusing mechanism, then the data near that jump is probably garbage, because the OPD was varying substantially during single integrations. Parameters are ΔOPD which is the allowable maximum ΔOPD (microns), jumpOpd which is the maximum allowable jump (microns) and sideDrop which is the number of points to drop on each side of a jump. The current defaults are:

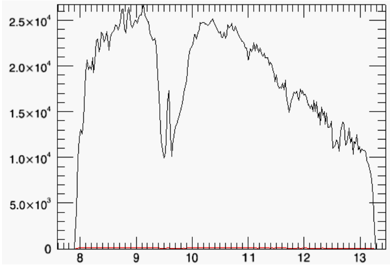
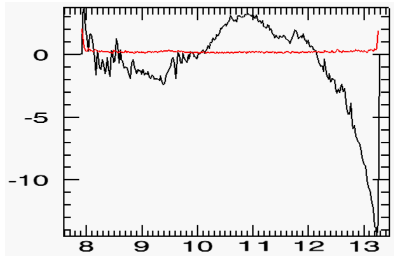
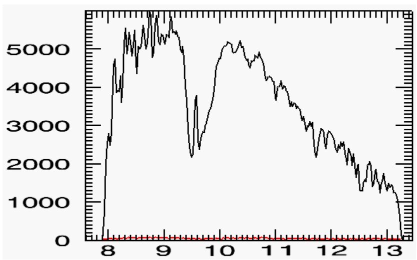
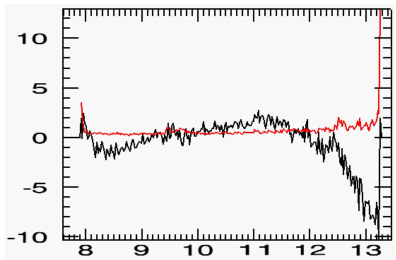
ΔOPD = 150 microns (PRISM)
ΔOPD = 800 microns (GRISM)
jumpOpd = 10 microns (from one frame to the next)
sideDrop = 1

The output contains a FLAG table as described in the original OIR FITS document, and is essentially a list of times that contain data flagged as no good, and the reason why:

TAR: 54207.344 54207.345 1 1 -1 1 -1 GROUP - TRACKING DELAY TAR: 54207.344 54207.344 1 1 -1 1 -1 JUMP IN GROUP DELAY TAR: 54207.344 54207.344 1 1 -1 1 -1 JUMP IN GROUP DELAY TAR: 54207.344 54207.344 1 1 -1 1 -1 JUMP IN GROUP DELAY TAR: 54207.344 54207.344 1 1 -1 1 -1 JUMP IN GROUP DELAY TAR: 54207.344 54207.344 1 1 -1 1 -1 JUMP IN GROUP DELAY TAR: 54207.344 54207.344 1 1 -1 1 -1 JUMP IN GROUP DELAY TAR: 54207.348 54207.348 1 1 -1 1 -1 END of OBS	***target.flag.fits*** ***calibrator.flag.fits***	flag table *[text file]*

2.2.7. Coherent Average

Finally all unflagged but phase rotated frames are averaged together (sum in horizontal direction) to form the simple estimated **correlated flux** (unnormalized complex visibility) by the routine **oirAverageVis**. The output is in the OI_VISIBILITY format and inhabits also the **correlated phase** table. Besides it contains also the **wavelength table** which maps the detector pixel scale to the wavelength scale in µm.

		***target.corr.fits*** correlated flux with error *[in adu/s]* and correlated phase with error *[in deg]* for the target
		***calibrator.corr.fits*** correlated flux with error *[in adu/s]* and correlated phase with error *[in deg]* for the calibrator

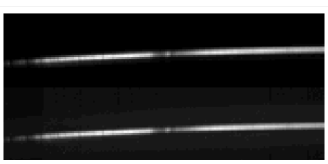
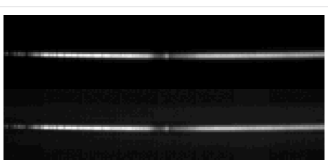
2.3. Normalization: Instrumental Visibility, Phase and Flux

What is with compression and editing?

Now, it re-runs **oirFormFringes** on the same (uncompressed) interferometric data to get again the fringe image data structure like in 2.2.2. The main difference is now that it is done for each interferometric channel (I1 and I2) separately (output: *target/calibrator.fringes1.fits* and *target/calibrator. fringes2.fits*), so no removel of the backgound by taking advantage of the 180 degree phase difference between both interferometric channels.

Also the alignment (de-rotation, 2.2.5.) with **oirRotateGroupDelay** is done again, but also separately on each channel for the fringe image data structure using the instrumental OPD and estimated group delay from 2.2.4. The output file *target/calibrator.fringemap1.fits* and *target/calibrator.fringemap2.fits* is deleted (cleaned up) at the end if the keyword “no delete” is not set.

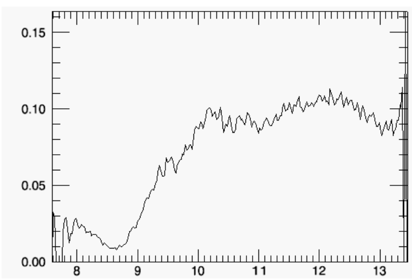
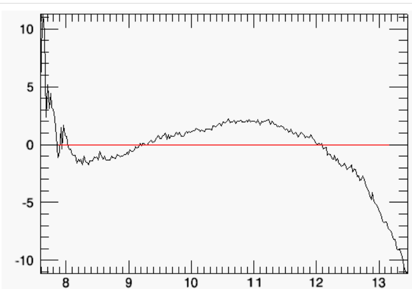
With the routine **oirMeanRMS** one gets finally, pixel by pixel, a 2-D MAP of the **correlated flux** for the I1 and I2 channel by computing the mean and RMS of the previous output.

	***target.RMS1.fits*** ***calibrator.RMS1.fits***	real and imaginary part of the correlated flux map for channel I1 *[2 element array of structures with DATA1 (342x40 elements)]*
	***target.RMS2.fits*** ***calibrator.RMS2.fits***	real and imaginary part of the correlated flux map for channel I2 *[2 element array of structures with DATA1 (342x40 elements)]*

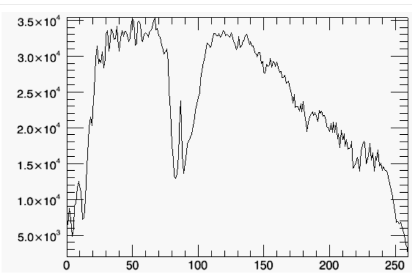
The grand finale is now done channel by channel, pixel by pixel, for I1 and I2 separately in the routine **spVis**. It fits the **estimated SQRT(PA*PB)** (*target/calibrator.geo.fits* or *target/calibrator.geo2.fits*) to the **estimated correlated flux** (*target/calibrator.RMS1/2.fits*) to get the instrumental visibility:

instrumental visibility = correlated flux / SQRT(PA*PB) (for I1 and I2 first)

In other words it finds the number which when multiplied by the geometric y-profile for a specific channel, best fits the correlated signal profile in the same channel (with allowance for a linear background in the geometric signal). This number is the instrumental visibility for this channel. It then averages the I1 and I2 results, and produces the instrumental visibility and phase.

<table>
<tr><td>
</td><td rowspan="2">***target.insvis.fits***
calibrator.insvis.fits</td><td>instrumental visibility</td></tr>
<tr><td>
</td><td>instrumental phase *[in deg]*</td></tr>
</table>

The estimation of the instrumental photometry is then done in the following way: the measured PA+PB signals in channel I1 and I2 (in *target/calibrator.ABchop.fits*, I1_A + I1_B and I2_B + I2_A ?) is processed using the estimated PA+PB PSF in these channels (in *target/calibrator.ABphotometry.fits*). The average is then outputad as a single instrumental photometric array.

<table>
<tr><td>
</td><td>***target.rawphot.fits***
calibrator.rawphot.fits</td><td>instrumental flux *[in adu/s]*</td></tr>
</table>

3. Calibration

===========

Routine: **spCalVis, target, calibrator**, diam=diam, flux10=flux10, calspec=calspec

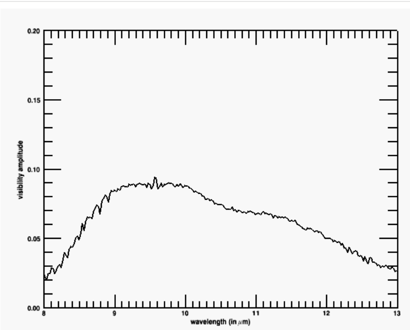
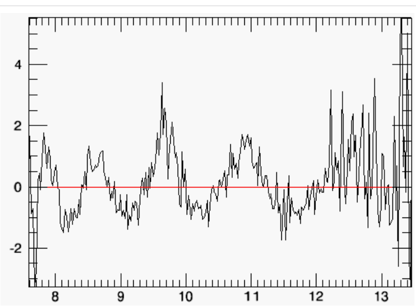
With this routine one gets the calibrated visibility amplitudes, phases and fluxes for the target from the instrumental quantities of the target and calibrator.

The calibrated visibility amplitude are just the instrumental value (as a complex phasor) divided by the value for the calibrator. If the calibrator is somewhat resolved the diameter can be specified as diam=diam (in mas) and a uniform disk model will be applied during the calibration.

calibrated visibility = insvis (*target*) **/ insvis** (*calibrator*) **/ theovis** (*uniform disk*)

The calibrated phase (in deg) is the target instrumental phase minus the calibrator instrumental phase (modulus 360 degree) since the calibrator is to be assumed phase less.

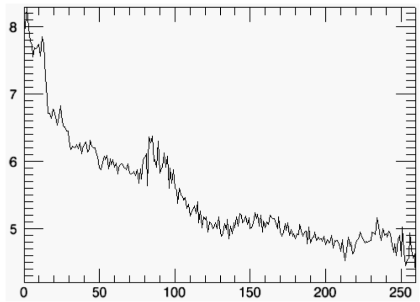
calibrated phase = insphase (*target*) **- insphase** (*calibrator*)   (mod 360 deg)

	***target.calvis.fits***	calibrated visibilities *[VISIBILITY table]*
		calibrated phase *[in deg]* *[VISIBILITY table]*

The calibrated flux (in Jy) under the mask is an estimation of the photometric flux of the calibrator, the inputed flux10 (Jy) value of the calibrator and a blackbody spectrum:

calibrated flux = flux10 * (10 / λ)$^{-2}$ * rawphot (*target*) **/ rawphot** (*calibrator*)

Default assumption is that the calibrator looks closely enough like a Rayleigh-Jeans blackbody (S ~ λ^{-2}) in the N-band, and that its flux at 10.0 microns is the value specified in flux10. Another approach is that if one have the correct spectrum of your calibrator e.g. ISO data, one should specify 1 Jy as flux10 and scale up the photometric output by hand to the correct values.

	***target.calphot.fits***	calibrated flux *[in Janskys]* *[DATA table]*

4. Additions

==========

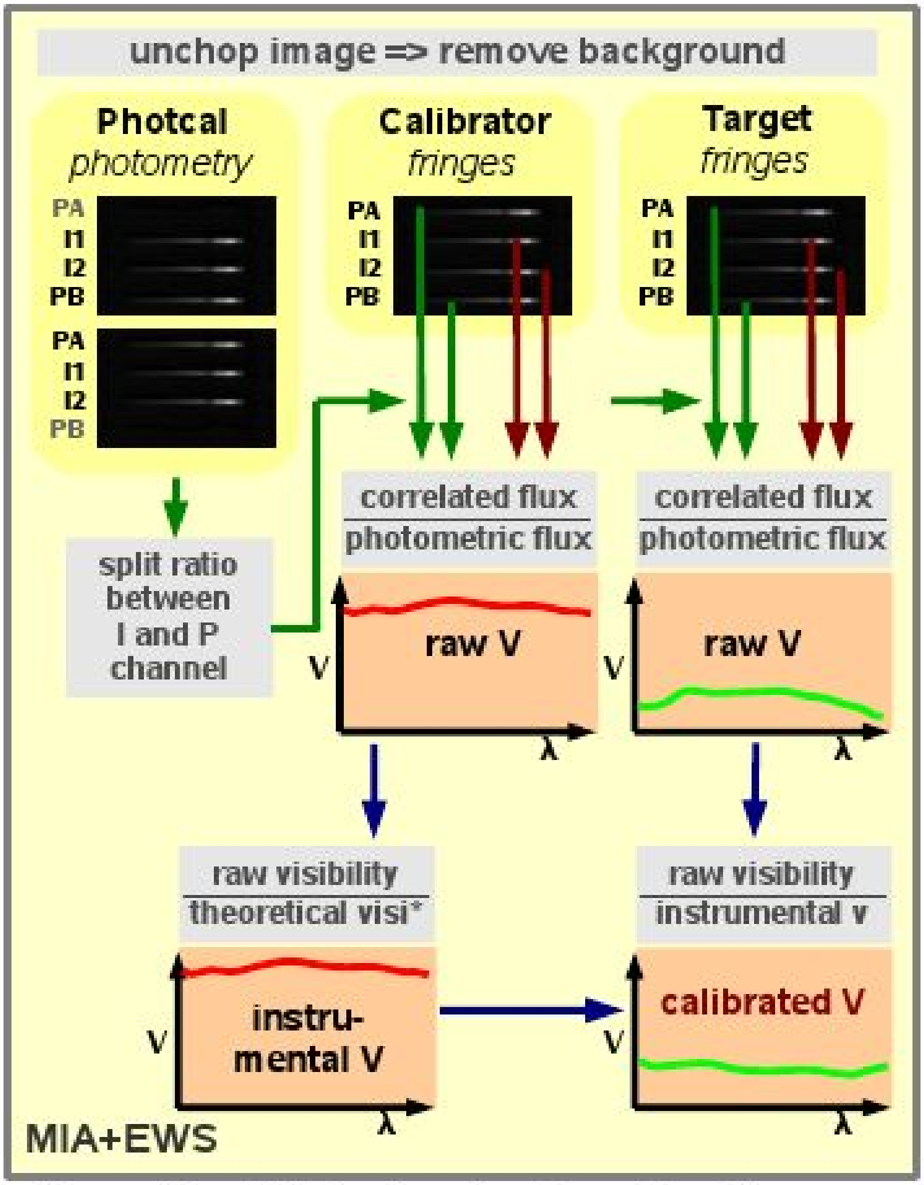
4.1. Errors and Signal to Noise Ratios

Since the determination of any errors is very difficult throughout the reduction one has to estimate the errors in other ways. While for HIGH-SENSE observations the typical visibility error is in the order of 10 %, SCI-PHOT observations should have a much smaller error (because correlated and uncorrelated flux is observed simultaneously and the kappa coefficients should be relatively stable). The error can be approximated by comparing the result of different target calibrations with different calibrators for the according night.

The approximate needed source flux to get a certain signal to noise ratio (S/N):
(also true for the correlated flux / for visibility = 1)

	UT	**AT**
noise level:	300 mJy	6 Jy (20xUT)
1 % S/N	30 Jy	600 Jy
10 % S/N	3 Jy	60 Jy

4.2. Sketch of the Reduction



*from uniform disk for given diameter and baseline