VLTI School Practical Sessions

September 2013

Constraints on stellar kinematics using spectro-interferometry
Application to a Be star disk

Spectro-interferometry is a powerful technique combining high spectral and spatial resolution. It allows not only to constrain an object geometry as "standard interferometry", but also its kinematics by centering observations on some narrow spectral features sensitive to the Doppler effect.

During this practical session we will focus on the case of classical Be stars observed in some emission lines. We will analyze some VLTI/AMBER data with a simple kinematic-model developed in IDL to probe the circumstellar environment of this class of objects.

Remember that, beyond this example, such technique can be applied to all kind of objects with absorption or emission lines in their spectra : stellar photospheres, young stellar objects, novae, interacting binaries, and even AGNs!

1 Some useful background

1.1 Spectral lines in stellar spectrum

1.1.1 Stellar spectrum

A stellar spectrum can always be separated in two kind of features : continuum and discrete spectra. The most common continuum spectra is the black-body, which is emitted by any body at a non-zero thermodynamic equilibrium. This emission usually dominates stellar spectra. Nevertheless, other kinds of continuum spectra can be found in specific physical conditions : free-free (also called Bremsstrahlung), free-bound, synchrotron, or cyclotron emissions.

The most common discrete features in stellar spectrum are spectral lines. They correspond to a photon emitted or absorbed by a quantum system such as an atom or a molecule. Consequently, their wavelengths are determined by the possible energy levels for the considered particle. Depending on the physical conditions in the medium and the intensity of the continuum radiation on which they formed, spectral lines can be found in absorption or emission. Absorption lines are usually formed in stellar photospheres, whereas emission lines are often found in extended diluted medium highly irradiated by a central source, i.e. in circumstellar environments. Note that other kinds of discrete spectra exist, such as spectral bands which stem from the merging of many close-by spectral lines for complex polyatomic systems. (molecular bands).

1.1.2 The hydrogen lines

During this practical session we will focus on the most abundant element in the universe : hydrogen. The wavelength (λ) of a photon emitted by a transition between two levels of this atom is given by the Rydberg-Ritz formula :

$$\frac{1}{\lambda} = R_H \left(\frac{1}{(n + \Delta n)^2} - \frac{1}{n^2} \right) \tag{1}$$

where R_H=1.097m^{-1} is the Rydberg constant, n the ground level of the transition defining the serie to

which it belongs to, and Δn, the level difference defining the position in the series. Note that both n and Δn are strictly positive.

Question : The first six series of the hydrogen atom are named after their discoverers. Can you name them ?

The hydrogen lines are almost always named after their series and their position in them using a Greek letter. X_α correspond to a transition between n_X and n_X+1, where n_X is the ground level of the X series. X_β corresponds to a transition between n_X+2 and n_X... Finally X_∞ corresponds to the theoretical limit of the series when $\Delta n \rightarrow \infty$.

Question : Gives the wavelengths of X_α and X_∞ for the first six series ?

Question : Which hydrogen lines are observable with AMBER (H and K bands) ? What about VEGA (R and I) ?

1.2 Doppler effect

1.2.1 General concept and application to electromagnetic waves

The Doppler effect is the shift in frequency of any kind of periodic event for an observer moving relative to its source. This applies to all kind of waves and in the case of an electromagnetic wave the shift in wavelength is given by :

$$\delta\lambda = -\frac{\mathrm{v}}{c}\lambda_0 \tag{2}$$

where λ_0 is the rest wavelength, v the radial velocity between the source and the observer, and c the speed of light in the vacuum. If the source and observer are moving away, i.e. v is negative, the wave is red-shifted, whereas if they are coming closer the wave is blue-shifted.

Question : What would be the shift for a $2\,\mu$m radiation with a radial velocity of $300\,\mathrm{km\,s^{-1}}$?

1.2.2 Effects on spectral lines

The Doppler shift affects both continuum and spectral lines. However, for non-relativistic radial velocities between the source and the observer, the effect on broad continuum emission such as a black-body is often unnoticeable as the variation of the emission intensity between the rest wavelength and the shifted one is usually negligible. On the other hand, the effect can easily be detected on spectral lines as their natural width are very small.

The effect of Doppler Shift on spectral lines can be divided into two types. The first one is a global shift of the line wavelength due to some global radial velocity between the source and the observer. Such velocity can be due to earth motion around the sun (i.e., see heliocentric correction), to stellar motion in a binary system or around the galactic center, or to cosmological redshift introduced by the expansion of the universe.

On the other hand, a velocity distribution in the emitting medium causes a line deformation as the different parts of the emitted light can be blue or red-shifted. This distribution, that can be either microscopic or macroscopic, often causes a line broadening.

The most common microscopic velocity distribution is due to the kinetic temperature. For an ideal gas the rms-velocity of particles is given by:

$$v_{rms} = \frac{3kT}{m_p} \tag{3}$$

where k is the boltzman constant (k=1.38 10^{-23} m^2 kg s^{-2} K^{-1}), T the particles kinetics temperature, and m_p the mean particle mass.

The most common macroscopic effects on the velocity distribution are due to stellar rotation, turbulence, and expansion in a stellar wind.

Question : Is the line broadening dominated by kinetic temperature or stellar rotation for the sun (T_{eff} = 5700K, $v_{rot} \sim 2$ km $^{-1}$) and for the Be star α Col (T_{eff} = 13000K, $v_{rot} \sim 400$ km $^{-1}$) ?

1.3 Classical Be stars

Classical Be stars are hot non-supergiant stars that have at least exhibited once the so called "Be-phenomenon", i.e. emission lines and IR-excess in their spectra originating from a dense gaseous circumstellar environment highly illuminated by the star.

A generally accepted scheme is the presence of two distinct regions in their environment : a dense equatorial disk dominated by rotation and responsible for most of the line emission and IR-excess and a more diluted radiatively driven polar wind with terminal velocities on the order of 1000km.s^{-1}.

However, the physical process or processes responsible for the mass-ejection and reorganization of matter in the circumstellar environment are still highly debated. The effect of rotation, radiative pressure, pulsation, and binarity have still to be quantified.

An efficient way to test the various hypothesis on the mass ejection is to constrain both the geometry and kinematics of their circumstellar environment. However, considering their typical extension, i.e. a few milli-arc-seconds, this can only be done using spectro-interferometric measurements.

2 The kinematic model

2.1 Description of the model and parameters

In this practical session we will use a "toy" model simulating the emission from a geometrically thin rotating and/or expanding equatorial disk. The model computes a serie of narrow-spectral-band images through an emission line with the considered spectral resolution. It is described in details in Delaa et al. (2011).

In this model the emission from the star $I_\star$(x,y) is modeled as a uniform disk, and the envelope emission in the continuum I_c(x,y) and in the emission line I_l(x,y) as two elliptical Gaussian distributions with a flattening due to a projection effect of the geometrically thin equatorial disk, i.e., $f = 1/cos(i)$, where i the the object inclination angle. The radial and azimuthal velocities in the disk are given by :

$$V_{\rm r} \;=\; V_0 + (V_\infty - V_0)\left(1 - \frac{R_\star}{r}\right)^\gamma \tag{4}$$

$$V_\phi \;=\; V_{\rm rot}\left(\frac{r}{R_\star}\right)^{\beta}. \tag{5}$$

From these two velocity distributions a projected velocity map along the line of sight is computed using the following projection formula:

$$V_{proj}(x,y) = \left(V_\phi \, sin\,\phi - V_r \, cos\,\phi\right) sin\, i \tag{6}$$

For each spectral channel considered in the line, an iso-velocity map projected along the line of sight is then calculated and multiplied by the whole emission map in the line. Finally, the whole emission map for each wavelength consists of the weighted sum of the stellar map, the disk continuum map and the emission line map within the spectral channel under consideration. The map is then rotated by the major axis position angle (P.A.), and scaled using the stellar radius ($R_\star$) and distance (d). The complete chart of the algorithm is presented in Fig **??**.

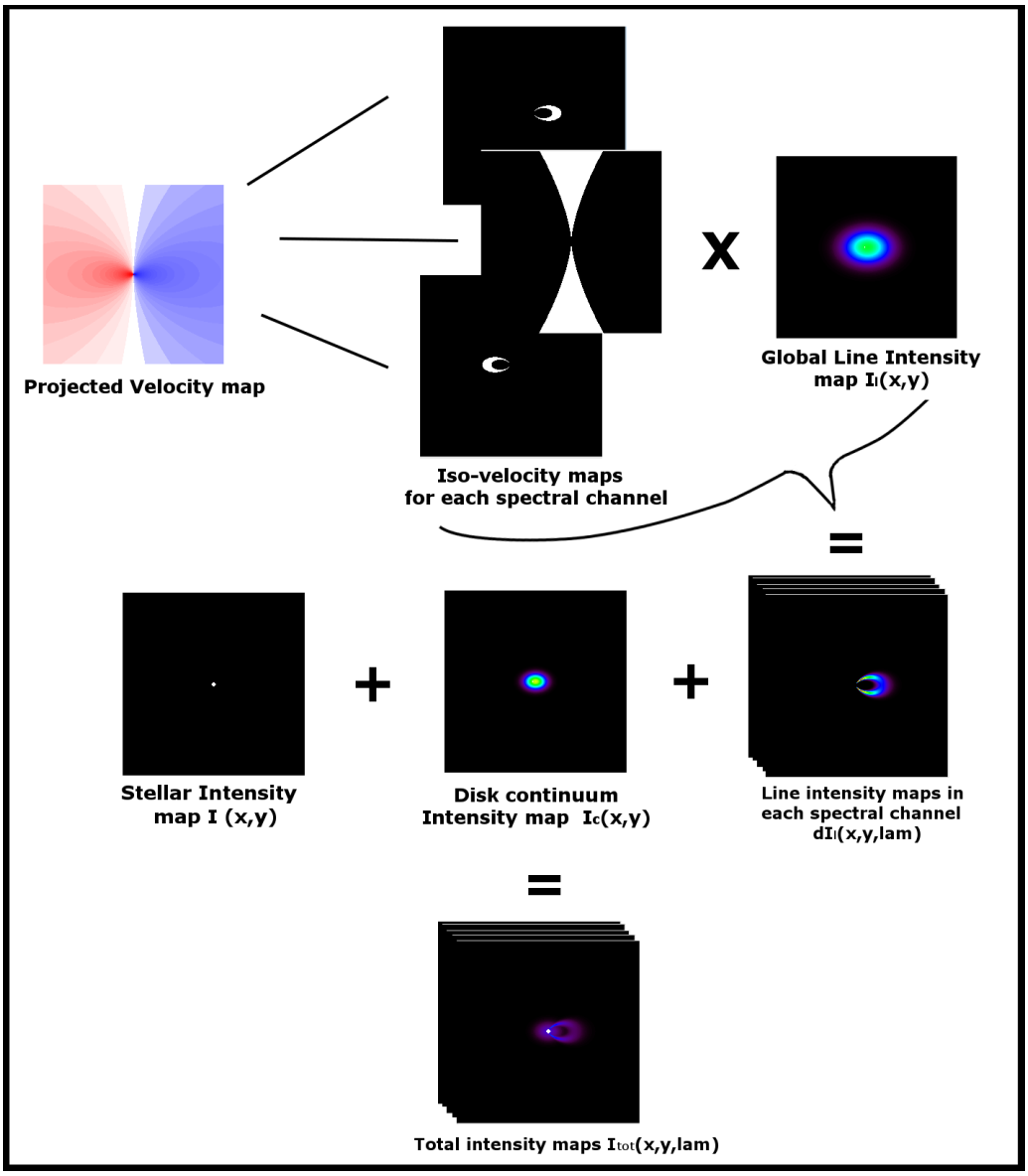

Figure 1: Schematic representation of the kinematic model operation.

The model parameters can be classified into 5 categories:

1. The global model parameters : size of the simulation in pixels (n_x=n_y), field of view in stellar diameter (fov), number of wavelength (n_λ), central wavelength of the emission line (λ_0), step in wavelength ($\delta\lambda$), and spectral resolution (R).

2. The global geometric parameters: stellar radius ($R_\star$), distance (d), inclination angle (i), and disk major-axis position angle (PA).

3. The disk continuum parameters: disk FWHM in the continuum (a_c), disk continuum flux normalized by the total continuum flux (F_c).

4. The disk emission line parameters: disk FWHM in the line (a_l) and line equivalent width (EW).

5. The global kinematic parameters: stellar rotational velocity (V_{rot}), expansion velocity at the photosphere (V_0), terminal velocity (V_∞), exponent of the expansion velocity law (γ), and exponent of the rotational velocity law (β).

The model final product is a $n_x \times n_y \times n_\lambda$ data-cube saved in fits format and composed of n_λ narrow-bands $n_x \times n_y$ pixels images.

As it is now widely admitted that the classical Be star equatorial disk are dominated by rotation (i.e. $V_\phi >> V_\theta$), we will only consider in this practical session the case of rotating disk. Thus all the models we will run will have V_0=V_∞=0.

2.2 Creating your first rotating disk

Let's compute our first model. First launch IDL (*idlde*) in a terminal. The procedure that computes the kinematic-model, i.e. *Be_disk_model*, reads the input parameters from an ASCII file. You can find a prototype of such file at the following path at : *$HOME/spectro-interf/param_example.dat*

Open this file using your favorite text editor (Gedit or Emacs for example). You can see that it contains all the model parameters defined in the previous section. We are going to use the current parameters for our first model.

First create a *dir* variable that will contains the path to our working directory :

dir="$HOME/spectro-interf/"

then an *inputfile* variable for the input file:

inputfile=dir + "param_example.dat"

and an *outputfile* variable that will contains the output datacube in fits format :

outputfile=dir + "datacube.fits"

Finally, to launch our first model we need to type :

Be_disk_model,inputfile,outputfile=outputfile,outputarray=map,
lam=lam,kinemap=kinemap,lineprofile=lineprofile

The output datacube will be copied both in the *outputfile* file and in the variable *map*. *lam* will contain the output wavelength table of the n_λ images, *kinemap* the projected velocity map, and *lineprofile* the emission line profile corresponding to this model.

You can check the size and type of each variable using the *help* procedure. For instance, if you type *help,map* IDL will tell you that map is a 256×256×101 float array, as define in the input file by the first and third parameters.

To print some variable values, just use the *print* procedure. For example type *print,lam* to see the values of 101 wavelengths for which an intensity map has been computed. This wavelength table was built using the n_λ, λ_0, and $\delta\lambda$ parameters.

Question : Transform the wavelength table into velocity using the previously defined Doppler-shift equation. What is the maximum Doppler-shift computed by this model?

2.3 Visualizing the outputs

Now that we have computed our first model let's plot the result. IDL offers two plotting procedures :

- *plot, x, y* to plot *y* as a function of *x*, according that they are both 1D-arrays of the same size. Use *oplot, x, y* to overplot another function on the same plot.
- *tvscl, array* to plot a two-dimensional array, i.e., an image.

After checking that they have the same dimension and size, plot the emission line profile as function of the wavelength using the plot procedure.

Question : A What is the shape of the line profile ?

Question : Are the min and max wavelengths dominated by continuum or line emission?

Question : Give the wavelengths (the two of them) corresponding to the maximum of emission? Find the corresponding indexes in the *lam* array.

Now let's try to plot some 2D-array using the *tvscl* procedure. *kinemap*, the disk projected-velocity field, is of this kind. You can plot it directly.

Question : Can you find from *kinemap* where are the red-shifted, the blue-shifted, and the unsifted part of the disk? Explain why!

Unfortunately *map* is a 3D-array (i.e., a data-cube) so we cannot plot it directly using *tvscl*. We first need to extract a 2D-slice from it. To extract the i^{th} xy-image from it do:

mapi=map[*,*,i]

The * symbol tells IDL to extract all elements from a vector. So *mapi* contains all x and y elements of map for the i-th value of λ. We will use this to plot images at different wavelengths inside (and outside) the emission line from our data-cube *map*.

Question : To begin our journey through the line, plot an image that correspond to the continuum emission. What do you see?

Change the image dynamics by plotting the square-root (using *sqrt* function or power operator ^) of this image.

Question : Do you see something new? Can you explain why the contrast between this two structures is so high?

Let's now plot the image corresponding to the center of the emission line.

Question : What is the shape of the disk emission at this wavelength? Is it compatible with what you deduced from the *kinemap* plot?

Plot the images corresponding to the maximum of emission in the line, and some images in the wings.

Question : What are their shape?

Finally, we will plot an animation of all the images in our data-cube. To do so, we will execute a loop using the *for* statement on all wavelengths indexes. We'll used a 0.3 power-law to enhance the image dynamics:

```
for i=0,100 do tvscl,map[*,*,i]^0.3
```

It might have been a bit fast, you may retype this a few times. Thanks to this animation you can follow the shape of the emission in all narrow spectral channels through the line.

Question : Where does the most shifted (in wavelength) emissions comes from? Explain this!

Question : What are their extension compared to the one of the maximum of emission?

3 Simulating spectro-interferometric observations

3.1 The simulator of observations

Now it's time to talk about interferometry. In this section we'll learn how to extract wavelength-dependent visibility and differential phase from our model. To do so we'll use the following procedure :

```
simulate_obs,outputfile,B=B,angle=angle,visi=visi,phase=phase,lam=lam
```

The main input of this procedure are the string variable *outputfile* containing the path of a previously computed data-cube stored in fits format, an array of baselines lengths *B* in meters, and an array of the same size *angle* filled with baseline orientations in degrees. The output are the wavelength dependent visibility (*visi*), differential phase (*phase*) in radians and the wavelength table *lam* (equal to the one that we already got from the *Be_disk_model* procedure).

Let's start with two 50 m baselines parallel and perpendicular to the disk major-axis (East-West in our test model). First defined the arrays for the baseline length and orientation :

```
B=[50,50]
angle=[90,0]
```

Then just run the simulator. The output *visi* and *phase* are 2D-arrays with a dimension equal to 2×101. Their first row, i.e. *visi[0,*]* and *phase[0,*]*, corresponds to the visibility and differential phase for the first baseline, and their second row, to the same quantities for the second baseline.

3.2 Plotting the outputs : visibility, differential phase, and spectra

Use the *plot* procedure to plot the first baseline visibility as a function of the wavelength. Overplot the second baseline visibility using *oplot*. To differentiate the two curves, you can set the line style and/or color of one of them using keywords *linestyle=* with value 0 (solid), 1 (dotted)... and *color=* (with a value from 0 to 255 with color depending on the already loaded palette). Try :

```
plot,lam,visi[0,*]
oplot,lam,visi[1,*],linestyle=1,color=230
```

Question : Compare these two curves. What are their differences? Explain them using your knowledge about the visibility and remembering the narrow-spectral channels images that you plotted in Sect. 2.3

Let's plot the differential phase for the two baselines but in degree instead of radian. To do so we can use either the idl variable *!DPI*=π or *!RADEG*=180/π. A differential phase is always given between -180^o and +180^o. We can specify and x or y range for the plot using the keywords *xrange=* and *yrange=*. Try :

```
plot,lam,phase[0,*]*!RADEG,yrange=[-180,180]
oplot,lam,phase[1,*]*!RADEG,color=230
```

Question : Describe the phase variation through the line for both baselines? Once again, explain their differences.

You might have noticed that the line profile was computed by *Be_disk_model* and not by *simulate_obs*.

Question : Can you find the reason why?

3.3 Effect of the baseline length

Here, we will study the effect of baseline length on the spectro-interferometric quantities. Define a new set of 10 baselines aligned with the major-axis with increasing length :

```
B=[10,20,30,40,50,60,70,80,90,100]
angle=[90,90,90,90,90,90,90,90,90,90]
```

The length and orientation could also be written in a much shorter way :

```
B=findgen(10)*10+10
angle=replicate(90,10)
```

The function *findgen(N)* returns a N-elements float-vector with values equal the indexes (i.e., 0 to N-1). The *replicate(val,N)* function also creates a N-elements vector with all values equal to *val*.

Run *simulate_obs* with these baselines and plot their visibility. You can use *oplot* in a loop to avoid rewriting 9 times the function :

```
plot,lam,visi[0,*]
for i=1,9 do oplot,lam,visi[i,*],color=50+20*i
```

Question : What is the evolution of the visibility with the baseline length in the continuum? And in the line ? Is that what you expected ?

Now let's plot all the differential phases with a y-range between -50 and +50 degree.

Question : How does it behave with the baseline length?

The IDL function *max(array)* returns the maximum value of *array*. We can use this function to extract the phase variation amplitude and plot it as a function of the baseline length. First we need to define float array explicitly using *fltarr(N)*. Then, we have to fill each element of the array with the maximum phase for a given baseline. We'll do that with a loop.

```
phase_ampl=fltarr(10)
for i=0,9 do phase_ampl[i]=max(phase[i,*])
plot,B,phase_ampl
```

Question : How does the phase behave with the baseline length? I know it's the same question that before, but you can now be a bit more quantitative...

Finally, compute and plot the visibility and phase variation for a set of 10 longer baselines(i.e., between 100 and 200 m).

Question : Are your previous observations still valid? Try to explain why.

3.4 Effect of the baseline orientation

From Sect. 3.2, we already have a hint of the effect of orientation on the qualitative morphology of the visibility and phase variations. To go further just define a set of baselines with a fixed length of 50 m and orientation going from 0 to 180^o with a step of 10^o.

Once again (and it's not the last time) compute and then plot the visibility and differential phase as the function of the wavelength for this new set of baselines.

Question : What is this effect of the baseline orientation on the interferometric measurements?

Finally, extract the maximum of phase for each baseline and plot it as a function of its orientation.

Question : What function does it look like? You can try to overplot such function if you have time.

4 Play with the model Parameters

Now we will start changing various model parameters and see their individual effects on the visibility and phases. To avoid waste of time by typing all the needed IDL commands (i.e., compute models, extract and plot visibilities and phases...) again and again, we will use a script that we'll run each time we change a parameter value.

4.1 Simplify your life with scripts

First go to the "File" menu and choose "Open a file". Choose the "be_test1.pro" file in the "home/spectro-interf/" directory. It opens in the IDL development environment.

You can see that you almost know all commands in this IDL script, except *!P.multi* which divide a window into sub-windows for plotting and *window* that open a new plotting window with a *xsize* and *ysize* specified. The other commands are just the ones we played with before.

Compile the script and launch it using the two corresponding buttons in the upper launch-bar. The script runs a new model and then computes the visibility and phase for a triplet of 50 m baselines with orientation 0, 45, and 90^o. Finally it plots the visibility for the three baselines in the three upper sub-windows, and the corresponding differential phases in the three lower sub-windows.

4.2 Let's go!

Now, every time we will change a parameter, we will have to run the script again. Here is the series of parameters that you should test and their corresponding values (you can test other values if you want). After testing a parameter don't forget to put back the initial value given into brackets before testing the following one.

- Inclination angle (in ^{o}): 0, 80, -40 (40)
- Position angle of the major-axis (in ^{o}): 0, 45, 135, 180 (90)
- Stellar Radius (in $R_\odot$) : 3, 12 (6)
- Distance (in pc): 50, 300 (150)
- Disk major-axis FWHM in the continuum (in $R_\star$) : 1, 6 (3)
- Continuum disk flux : 0, 1 (0.5)
- Disk major-axis FWHM in the line (in $R_\star$) : 5, 20 (10)
- Line equivalent width (in Å) : 5, 20 (10)
- Stellar rotational velocity (in km s^{-1}) : 100, 300, -300 (500)
- Exponent of the rotational law (in km s^{-1}) : -0.3, -0.7 (-0.5)
- Spectral resolution : 5, 10, (1.8)

Question : What are the main effects of each of these parameters on the visibility and phase? Don't forget that the effect can depend on the baseline orientation!

5 Fitting real AMBER data

Now that you know the basics of the kinematic-disk model, we will use it to fit real AMBER data taken in High-Resolution (R= 12000) on a classical Be star.

5.1 The object : α Col

Our target is α Col (HR 1956, HD 37795) one of the closest (d=80± 2 pc) and brightest (m_V=2.6, m_K=2.8) classical Be star. The data that you will use were recorded in January 2010 and were published in Meilland et al. (2012, A&A, 538, 110). They consist of two measurements taken using two different triplets of baselines : A0-G1-K0 and DO-G1-H0.

Here are some information on α Col taken from literature that might be useful for the modeling :

Name	distance (pc)	T_{eff} (K)	v sin i (km s^{-1})	V_c (km s^{-1})	pol. angle (deg)	estim. $R_\star$ ($R_\odot$)	estim. F_{disk} (in the K band)
α Col	80±2	12963±203	192±12	355±23	109	5.8	0.25

5.2 Reading and plotting AMBER data under IDL

Let's look at the data. To read VLTI/AMBER data under IDL we will use a home-made procedure that will extract all useful information (baselines, visibility, phase...) from AMBER OI-fits file in a directory. It's prototype is the following :

amber_getall_night,datadir,data=data,nobs=nobs

where *datadir* is the AMBER data directory, *data* is a quite complicated structure array containing all information (baselines length, orientation, wavelength visibility, differential phase, spectra...) that will be extracted from the AMBER Oi-fits files and *nobs* is the number of observations (i.e. files) found in *datadir*. To define the path to our AMBER data folder just type :

datadir=dir+"amber/"

The members of the *data* structure are the following:

- *nlam* : the size of the wavelength table
- *lam* : the wavelength table (size=*nlam*)
- *nB* : the number of baselines (usually equal to 3)
- *B* : the baseline length (size=*nB*)
- *angle* : the baseline orientation (size=*nB*)
- *sqvis* : the square visibility (size=*nB*×*nlam*)
- *sqverr*: the square visibility uncertainty (size=*nB*×*nlam*)
- *phi* : the diffferential phase , (size=*nB*×*nlam*)
- *pherr* : the differential phase uncertainty (size=*nB*×*nlam*)
- *cphi* : the closure phase (size=*nlam*)
- *cpherr* : its uncertainties (size=*nlam*)

However, to allow flexibility on the dta format, all "array-type" members are actually pointers to the array. To access the corresponding array we must use the * operator. As an example, for the wavelength table for the first observation, just type :

***(data[0].lam)**

The corresponding square visibility for the first baseline of this first observation is given by:

(*(data[0].sqvis))[0,*]

Don't forget the two pairs of brackets when you want to access to a specific element in the array!!

You might have noticed that the spectrum is not a member of the data structure. This is because AMBER spectrum is not present in the final calibrated Oi-fits file. In this practical session we will use a spectrum extracted from an uncalibrated AMBER file saved in ASCII format in the file *alpha_col_final_spectrum.spec*. We will use the function *ReadArray* to read this file and extract a 2×*nlam* array containing the wavelength table in the first row and the calibrated flux in the second one.

Now we can plot the line profile and the visibilities and differential phases for all baselines and all observations with two loops. Open the file "Be_test2.pro" in IDL and look at the code. When you understand what it does (not before!!), compile it and run it. Note that the *xyouts* procedure is used to print the baseline length and orientation over the plot.

5.3 Qualitative analysis of the data-set

Before modeling the observations, we should start by a qualitative analysis of the data-set to determine if we can put some initial constraints on few of the parameters.

Question : Looking at the "V" or "W" shapes of the visibility drops in the line, can you determine the baselines close to the major-axis, minor-axis, and the intermediate ones?

Question : Consequently, can you roughly determine the disk major-axis orientation?

Now, look closer at third and fourth baselines. They roughly have the same length (i.e. B ≃ 90 m) and have very different orientation (-157.9^o and -88.9^o). They measure the disk extension (in the line and the continuum) at the same spatial frequency but in different orientations.

Question : Can you tell something about the object inclination angle?

From the table in Sect. 5.1 we know that 25% of the continuum flux comes from the disk and 75% from the star.

Question : Give the stellar diameter in mas assuming the distance and stellar radius given in that same table.

The function *udisk* simulates a uniform disk. It has four parameters : the visibility V , the wavelength *lam* (in μm), the diameter D (in mas), and the baseline length B (in m). Given three parameters, this function will return the value of the fourth one. For example :

V=udisk(lam=2.17,D=2,B=50)

will return the visiblity for a 2 mas uniform disk observed at 2.17 μm with a 50 m baseline, or :

D=udisk(lam=0.6,V=0.7,B=70)

the uniform-disk equivalent diameter for an object for which we obtain a 0.6 visibility at 0.5μm with a 70 m baseline.

Question : Is the stellar surface significantly resolved for our longer baseline?

The function *gdisk* is equivalent to *udisk* but for a Gaussian disk. The parameter *D* is replace by *fwhm*. for example,

fwhm=gdisk(lam=0.6,B=70 V=0.7)

return the Gaussian-equivalent FWHM for the same parameters.

Question : What is the size of the disk in the continuum (in mas and stellar radii)? You need to know how to compose visibilities for a not fully resolved object!

Question : Can you do the same in the line? Explain the limitations and try to give a roughly estimate of the disk extension.

5.4 Manual fitting process

Now that we have a few starting constraints on the kinematic-model parameters (from the table in Sect. 5.1 and our qualitative analysis in Sect. 5.3) we can start running models and comparing their results with the data. Modify the *param_example.pro* file using your first guess of the model parameters.

Then, open the script *be_test3.pro* and look at it. It is a mix a the *be_test1.pro* and *be_test2.pro* scripts. It reads the AMBER data, extract all the baselines length and orientation, and put them into two 1D-arrays, *B* and *angle*. It then computes a model using the *param_example.dat* file, and simulate the observations using the extracted baselines length and position angle. Finally observations are plotted the same way as in *be_test2.pro* and the simulated visibilities and differential phases are overplotted.

Run the script with your first guess of the parameters.

Question : Is the continuum well fitted for all baselines? What about the variations in the line?

With the work done in Sect. 4 you should roughly know what's the effect of the different parameters on the visibility and phase. If it's not already well-fitted, try first to adjust the level of the visibility continuum by changing the parameters having an effect on it. Then choose the parameters one-by-one starting with the one having a bigger effect a the visibility, phase and spectra. For example, you should determine easily the line Equivalent Width by adjusting the line profile.

Question : Now it's time to find a good set of parameters to fit the data... Good luck!

5.5 BONUS Section : Quantitative "goodness" of the fit with χ^2 computation

Fitting the data manually might be funny but it is not efficient and it can introduce human-based biases. The first step toward the implementation of automatic fitting techniques is to quantitatively measure our fit quality. The most simple way is to measure the distance between the data and the model for every measurement and compare it to the uncertainty on the measurement. The most common operator used for such purpose is the χ^2 defined as follow :

$$\chi^2 = \sum_{i=0}^{N} \frac{(D_i - M_i)^2}{\sigma_i^2} \tag{7}$$

where N is the number of measurements, and D_i and M_i represent the values of the data and the model, respectively. The χ^2 is usually normalized by the degree of freedom defined by N-L, where L is the number of free-parameters of our model. A reduced χ^2 of 1 roughly means that the mean departure between the model and the data is 1σ, a χ^2 of 4, that the average departure is 2σ, and so on. On the other hand χ^2 of less than 1, means that the model fits "too well" the data and that it has too many free-parameters.

5.5.1 Step 1 : interpolating the model wavelength-table

Before starting to implement the χ^2_r, we first need to interpolate our modeled quantities (visibility, phase, and spectra) at data wavelengths in order to be able to perform the subtraction between D_i and M_i. To achieve this task, we will use the IDL function *interpol*. To interpolate the jth baseline visibility and phase for the ith observation, type:

```
visi_interp=interpol(visi[i*3+j,*],lam,*(data[i]).lam)
phase_interp=interpol(phase[i*3+j,*]*!radeg,lam,*(data[i]).lam)
```

Note that we convert directly during the interpolation the modeled phase in degree as the observed one is given in that unit.

Question : How about the line profile? Find the IDL code to interpolate the model one at the observed wavelength?

You can implements the interpolated visibility, differential phase and spectra in the *be_test3.pro* script and over-plot them to check that you didn't do any mistake.

5.5.2 Step 2 : Estimating uncertainties on the measurements

The second step of the work is to estimate the uncertainties on the measurements. The AMBER Oi-fits files already contains the errors on the square visibility and differential phase. For the jth baseline of the ith observation, they are defined the following way :

```
sqVerr_ij=(*(data[i].sqVerr))[*,j]
pherr_ij=(*(data[i].pherr))[w,j]
```

You can extract the uncertainty on the visibility from the square-visibility one and print the average visibility and phase uncertainties for each baseline.

Question : What are the magnitude of the uncertainties on the visibility and the differential phase?

Now look at the plotted data and especially at the noise in the continuum.

Question : Do you think that the uncertainties given in the AMBER file are overestimated or underestimated?

In fact the uncertainty on the visibility is a composition of a differential uncertainty between the spectral channels that can roughly be measured using the noise in the continuum, and an absolute error, uniform on all spectral channels, mainly introduced by the calibration process. On the other hand, as the differential phase is a differential quantity, it's uncertainty is only differential.

Let's try to compute the uncertainties using the continuum noise. We must first define a range of indexes in the wavelengths table corresponding to the continuum. This can be done using the IDL function *where*:

```
lam_min=2.164e-6
lam_max=2.167e-6
w_cont=where( *(data[i]).lam lt lam_min or *(data[i]).lam gt lam_max)
```

Here, we selected the indexes corresponding to λ <2.164μm or λ >2.167μm, so roughly every wavelengths excluding the line itslef. The indices in *w* may depend on the observation *i* as the wavelength tables for the two AMBER files may be different. Hence, put this line of code into the *i*-loop.

Finally we can now compute the σ on the visibility and differential phase using the IDL function moment that return an array containing the first 4 moments of the distribution : mean, variance, skewness, and kurtosis. To get σ type :

```
mom_Vij=moment(sqrt((*(data[i].sqvis))[j,w_cont]))
sigma_Vij=sqrt(mom_Vij[1])
mom_phij=moment((*(data[i].phi))[j,w_cont])
sigma_phij=sqrt(mom_phij[1])
```

Question : Compare the differential uncertainties computed on the continuum noise and the one given in the AMBER Oi-fits. Conclude on the level of the absolute visibility uncertainties.

5.5.3 Step 3 : Computing the least-square in the line

First, we need to define the indexes corresponding to wavelengths in the emission line :

```
w_line=where( *(data[i]).lam ge lam_min and *(data[i]).lam le lam_max)
```

with *lam_min* and *lam_max* equal to the one previously defined for the continuum indexes *w_cont*.

We now have everything to compute the χ^2 on the visibility and differential phase for the i_{th} observations and the j_{th} baseline:

```
chi2vij=total((sqrt((*(data[i].sqV))[w_line,j])-visi_interp)^2./sigma_vij^2.)
chi2phij=total((phase_interp-(*(data[i].phi))[w_line,j])^2./sigma_phij^2.)
```

Note that the total function returns the sum of all elements for an array.

The final χ^2_r is the sum of the χ^2 on all visibilities and differential phases divided by the number of measurements (each spectral channel is considered as an independent measurements) minus the number of free-parameters (i.e. 11). You can load the final script, *be_test4.pro* and compile it. The final χ^2_r is printed at the end of the procedure.

Question : What about the closure phase and the spectra? If you still have time (you're really fast) modify the script to take into account these two quantities in the χ^2 computation