UNIVERSITY OF SYDNEY

# Manual: Making measurements

VERSION 1.0

*Author:*
Alison WONG

*Email:*
a.wong@sydney.edu.au

April 3, 2018

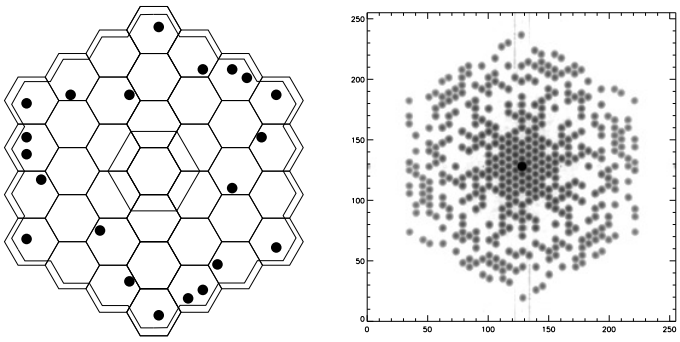
# Contents

# 1 Introduction

I have written the code to simulate taking interferometric differential polarimetric data from MCFOST models given a set of stellar parameters.

## 1.1 List of required files

- star.py
- physics.py
- run_MCFOST.py
- write_MCFOST_parafile.py
- Default_parameter class file



a b

Figure 2.2: 21 hole Golay mask for the Keck telescope (a) and the corresponding coverage of the power spectrum (b). (Tuthill et al., 2000)

# 2 An introduction to differential polarimetry

## 2.1 Interferometry and non-redundant aperture masking

Overcoming the limitations of atmospheric turbulence has been a perpetual challenge for astronomers. According to Rayleigh's Criterion, the angular resolution of a telescope with diameter $D$ at wavelength $\lambda$ should be $\lambda/D$, so a 10 m telescope, like the Keck telescope in Hawaii should have a resolution of 10 mas in visible light ($\lambda = 500$ nm). However, atmospheric distortion limits the resolution of all ground based telescopes to around 1 arcsecond (Labeyrie et al., 2006), the same as the fundamental resolution limit of a 10 cm telescope. This would not allow us to resolve even one of the stars with the largest apparent diameter, $\alpha$ Orionis (Betelgeuse), which has an angular size of $\sim 50$ mas, let alone give us the opportunity to probe its structure.

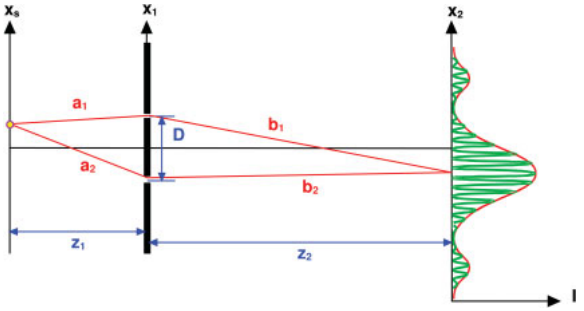



Figure 2.1: Fringe pattern produced from a single point source and a double slit aperture (Singer et al., 2005)

Stellar interferometry is a technique that has been used since the 1800s and has been a necessary device in the advancement of imaging technologies. It was first suggested by Fizeau (1868), who noted that the size of a source would affect the amount of smearing of the interference fringes it produced. Independent of Fizeau, Michelson invented the stellar interferometer and used it to determine the diameters of Jupiter's four largest moons (Michelson, 1891). This success led to the construction of the Michelson stellar interferometer at the Mount Wilson Observatory in California which was used to determine the diameter of $\alpha$ Orionis (Michelson and Pease, 1921).

The physics principles behind interferometry can be traced back to Young's double slit experiment (Young, 1802). When coherent light passes through an aperture of two slits, a fringe pattern is observed (Figure 2.1). This pattern arises from the optical path difference (OPD) that the light can take to reach the screen. The fringe contrast is called the *visibility* ($V$) and is defined in terms of $I_{\max}$ and $I_{\min}$, the maximum and minimum intensities in the fringe pattern.

$$V = \frac{I_{\max} - I_{\min}}{I_{\max} + I_{\min}}$$

In stellar interferometry the point source is replaced by a star and the pair of slits are replaced by two telescopes of separation $B$. A pair of such telescopes forms a baseline, and will have the equivalent resolution of a telescopes with diameter $B$. Further, it is the square of the visibility (called the power) which is recorded for a set of fringes, an observable that, with careful calibration, can be made somewhat robust to noise imposed by atmospheric seeing.

Rather than requiring a number of telescopes to produce a set of baselines, interferometry can be achieved with a single telescope using a technique called non-redundant aperture masking (Readhead et al., 1988; Tuthill et al., 2000). In non-redundant aperture masking a metal mask, such as the one in Figure 2a is placed over the pupil of the telescope. The mask is constructed with a number of carefully placed holes. Each pair of holes forms a baseline, and corresponds to a single sample of the power spectrum (Figure 2b). The mask is non-redundant if the vector separation of each pair of holes is unique. This ensures that the Fourier component of any baseline can be uniquely identified. If there was a redundancy

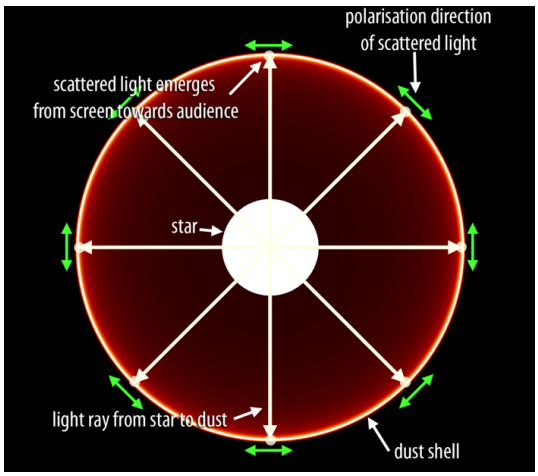



Figure 2.3: Diagram showing how light scattered off a dust shell is linearly polarised. (Norris et al., 2012)

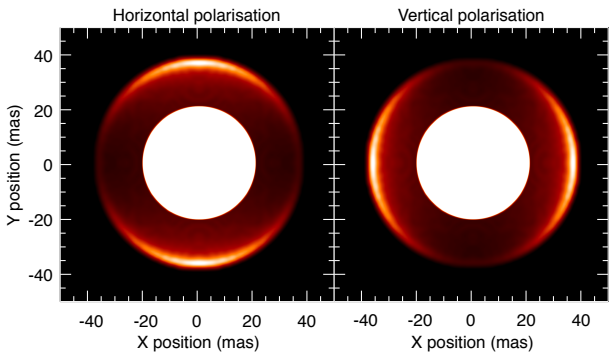



Figure 2.4: Modelled images of W Hya as seen through horizontally and vertically polarised filters. (Norris et al., 2012)

$R$ in a given baseline, then the corresponding power will add as a random walk with $R$ steps, and the power would be more subject to atmospheric noise and hence, more difficult to calibrate accurately (Tuthill et al., 2010).

## 2.2 Polarisation of light and Stokes formalism

Starlight is unpolarised, in that it is actually composed of light in many different and rapidly varying polarisation states. However, when this light is scattered off the dust shell it becomes partially linearly polarised (Figure 2.3). The simplest geometry of a star with a thin, spherical circumstellar shell imaged through a horizontally polarised filter will look larger vertically (Figure 2.4).

There are different formalisms to describe the polarised states but we will be using Stokes formalism. All forms of polarisation can be expressed using four stokes parameters: *I*, *Q*, *U* and *V*. Suppose that we had filters that allowed through linearly polarised light in the horizontal (*H*), vertical (*V*), $+45°$ (*A*) and $-45°$ (*B*) directions and we also had filters that allowed through right- (*R*) and left- (*L*) circularly polarised light. Then the Stokes parameters are the following:

$$I = H + V = A + B \qquad Q = H - V \qquad U = A - B \qquad V = R - L$$

Conversely, the different polarisation states can be represented in terms of the Stokes vector: $[I \quad Q \quad U \quad V]^T$

Horizontal $\mathcal{P}$-state ($H$) $\begin{bmatrix}1\\1\\0\\0\end{bmatrix}$ $\mathcal{P}$-state at $+45°$ ($A$) $\begin{bmatrix}1\\0\\1\\0\end{bmatrix}$ $\mathcal{R}$- state ($R$) $\begin{bmatrix}1\\0\\0\\1\end{bmatrix}$

Vertical $\mathcal{P}$-state ($V$) $\begin{bmatrix}1\\-1\\0\\0\end{bmatrix}$ $\mathcal{P}$-state at $-45°$ ($V$) $\begin{bmatrix}1\\0\\-1\\0\end{bmatrix}$ $\mathcal{L}$- state ($L$) $\begin{bmatrix}1\\0\\0\\-1\end{bmatrix}$

## 2.3 Differential polarimetry and the VAMPIRES instrument

The goal of differential polarimetry is to image polarised structures, which would otherwise be invisible due to the presence of bright, but unpolarised, sources. It has successfully been used to image dust around AGB stars (E.g. Ireland et al. (2005); Norris et al. (2012); Kervella et al. (2016)).

Traditionally, calibration is done using a reference star, but atmospheric and instrumental conditions change quickly with time. In differential polarimetry, calibration is done by comparing simultaneous observations through two orthogonal polarisation states either by taking the difference or the ratio. The advantage is that errors can be eliminated to obtain highly accurate observations.

The Visible Aperture-Masking Polarimetric Interferometer for Resolving Exoplanetary Signatures (VAMPIRES) is a new instrument, which combines interferometry with differential polarimetry (Norris et al., 2015). It operates in the visible/IR region (0.6 $\mu$m to 1.0$\mu$m), which is a shorter wavelength than other aperture masking instruments, exploiting the fact that light is more polarised at these wavelengths due to mechanisms involved in Rayleigh and Mie scattering. Although designed to image the inner most regions of protoplanetary disks, it has the capacity to image other polarised structures such as the dust shells around stars.

Differential polarimetry allows for observations to be made of faint polarised structures despite close proximity to very bright sources which would otherwise make them impossible to discern. This is combined with non-redundant aperture

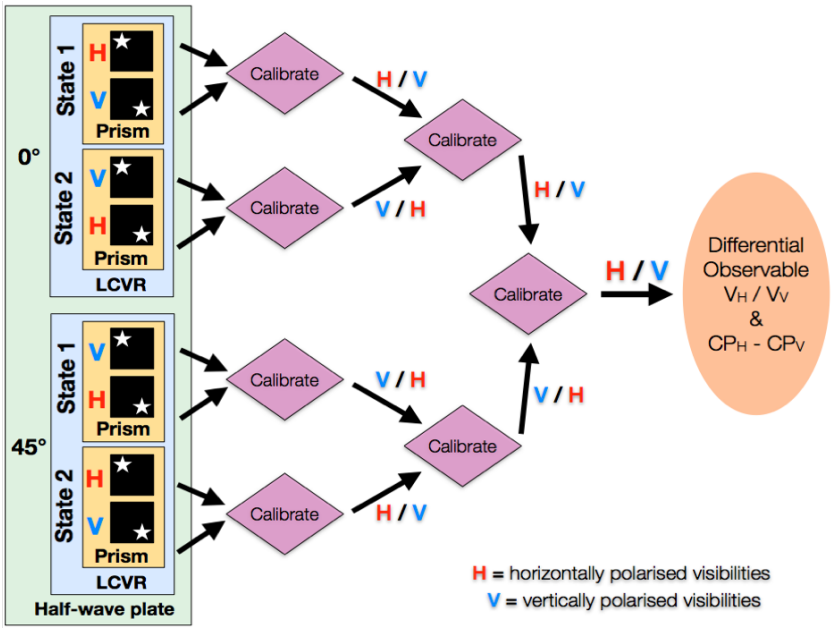



Figure 2.5: Diagrammatic representation of the 3-tier calibration in the VAMPIRES instrument (some details in text, for full details refer to Norris et al. (2015)). Obtained after calibration are the differential observables for the visibilities and closure phases although only the visibilities were used in this project. (Norris et al., 2015)

masking to attain the necessary resolution that cannot be achieved with differential polarimetry alone. VAMPIRES has a resolution of $\sim 10$ mas while other recent imaging polarimeters: Spectro-Polarimetric High-Contrast Exoplanet Research (SPHERE) (Beuzit et al., 2008) and the Gemini Planet Imager (GPI) (Macintosh et al., 2008) have resolutions of $\sim 100$ mas and $\sim 200$ mas respectively (Martinez et al., 2010).

VAMPIRES has a 3-tier calibration system (Figure 2.5). The first tier uses a Wollaston prism, which splits a light beam into two orthogonal polarisations, and allows them to be measured simultaneously. The second tier uses a Liquid-Crystal Variable Retarded (LCVR), which allows for fast switching between the two channels in the Wollaston prism ($\sim 10$ ms), and the third tier uses a half-wave plate, which rotates the polarisations by 45 degrees.

Calibration between two channels is accomplished by taking the ratio of the visibilities obtained from the two channels of the Wollaston prism are taken ($V_{\text{Ch }1}$ and $V_{\text{Ch }2}$). This is repeated with the polarisations rotated $90°$ by the LCVR ($V'_{\text{Ch }1}$ and $V'_{\text{Ch }2}$) and then calibrated against each other:

$$\left(\frac{V_{\text{Ch }1}}{V_{\text{Ch }2}} \Big/ \frac{V'_{\text{Ch }1}}{V'_{\text{Ch }2}}\right)^{\frac{1}{2}} = \left(\frac{V_{\text{Horiz}}}{V_{\text{Vert}}} \Big/ \frac{V_{\text{Vert}}}{V_{\text{Horiz}}}\right)^{\frac{1}{2}} = \frac{V_{\text{Horiz}}}{V_{\text{Vert}}}$$

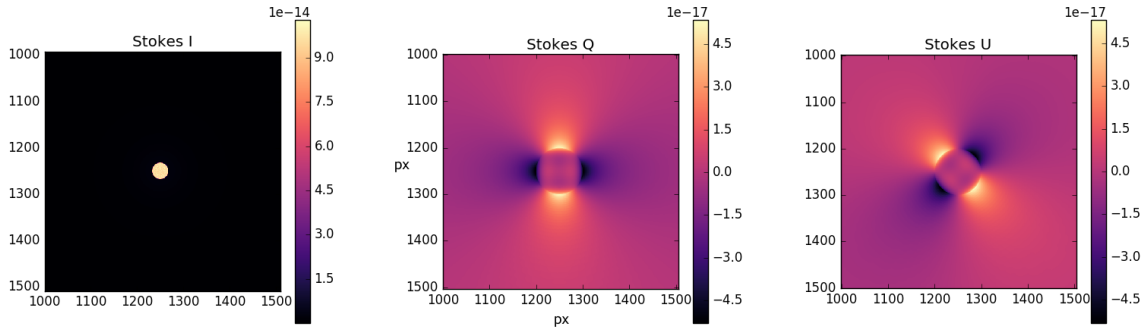
This is further repeated for the HWP, and for all other states as show in Figure 2.5. Calibration is done to remove instrumental noise that arises in different parts of the telescope, for example, non-common path error and imperfections of the LCVR. For full details, refer to Norris et al. (2015).

# 3 MCFOST

## 3.1 Making models: MCFOST

Stellar models were produced using the software MCFOST (Pinte et al. 2006, 2009). MCFOST runs a Monte Carlo radiative code used for modelling circumstellar environments. We produced 3 dimension envelope models using a spherical geometry. Scattering was calculated using Mie theory i.e. spherical grains. The inner and outer radius of the dust shell could be controlled and had a hard edge. It was assumed that both radiative and local thermal equilibrium were reached but the gas and dust were not in hydrostatic equilibrium. The star was modelled at a blackbody corresponding to the specified effective temperature. For a star centred at the origin, the image had vertical, point and axial symmetry. To create asymmetries, the star was offset from the origin while the dust remains centred. In this case there were no assumed symmetries.

MCFOST reads in properties of a star from a parameter file and observables are made using ray tracing methods. MCFOST outputted polarised images corresponding to the Stokes parameters I, Q, U and V. These images are produced using ray tracing methods. Note that only the Stokes I, Q and U images are required in this project. A typical set of images are shown in Figure 3.6 and the corresponding MCFOST parameters are given in Table. 1.

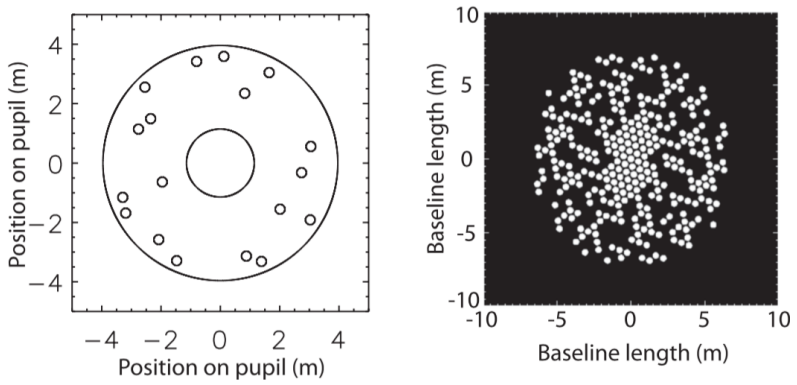



a b c

Figure 3.6: Example output of the Stokes I, Q, U images from MCFOST. The original map size was $2501 \times 2501$px but has been cropped to 1000:1500 in both axes. The inner radius of the dust shell is clear in the Stokes Q and U images, as a large amount of light is scattered here. The dust shell extends beyond the depicted region.

| Property | Value |
|---|---|
| stellar radius ($R_\odot$) | 150 |
| distance (pc) | 100 |
| temperature (K) | 3000 |
| mass ($M_\odot$) | 1.0 |
| dust mass ($M_\odot$) | 3.5E-6 |
| inner radius of dust shell (AU) | 2 |
| outer radius of dust shell (AU) | 300 |
| grid (px) | 2501 |
| map size (AU) | 150 |

Table 1: Some of the MCFOST parameter values used for the production of Figure. 3.6




a b

Figure 4.7: (a)18 hole nudged mask of VAMPIRES (b) corresponding coverage of the power spectrum. (Norris et al., 2015)

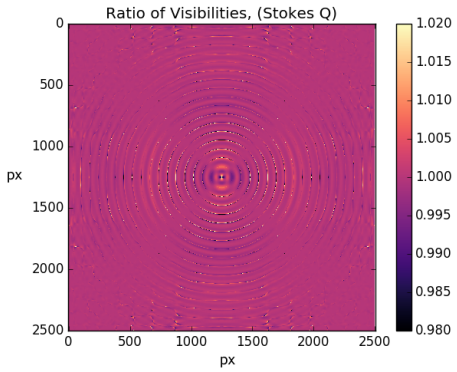



Figure 4.8: $PVR_Q$ corresponding to the images produced in Figure 3.6

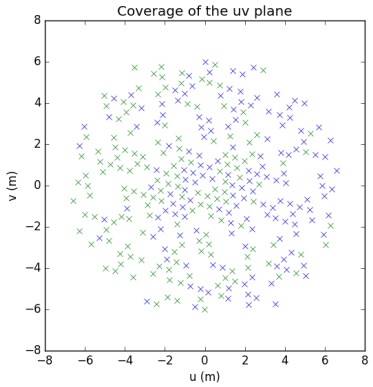



Figure 4.9: Coverage of the uv plane by an 18 hole aperture mask generating 153 baselines. Note that the coverage and PVR is inversion-symmetric as a pair of holes forms two baselines corresponding to the direction the vector of separation points but is counted as one since only one piece of information is recovered. This u,v coverage corresponds to the VAMPIRES 18 hole nudged mask (Figure 4.7)

# 4 Making measurements

### 4.0.1 Polarised visibility ratios (PVR)

Images of the horizontally and vertically polarised light ($H$ and $V$) and images of light polarised in the $+45°$ and $-45°$ ($A$ and $B$) can be obtained from the I, Q and U images as follows:

$$H = I + Q \quad V = I - Q \quad A = I + U \quad B = I - U$$

For each of these, we obtain the power spectrum (PS) by taking the absolute value of the square of the two dimensional Fourier transform.

$$PS(u, v) = |FFT2(\vec{x})^2|$$

These are then re-centered so that the 0 baseline (brightest point) is at the center and then normalised by dividing through by the maximum value. The two following polarised visibility ratios (PVR) corresponding to Figure 3.6 were produced.

$$PVR_Q = \frac{\text{normalise}(\text{center}(PS_H))}{\text{normalise}(\text{center}(PS_V))} \quad PVR_U = \frac{\text{normalise}(\text{center}(PS_A))}{\text{normalise}(\text{center}(PS_B))}$$

### 4.0.2 Sampling the PVRs

To produce data from the model that corresponds to that obtained from the telescope, the PVRs need to be sampled at points to achieve the same coverage as the aperture mask in the uv-plane; in this case, the VAMPIRES 18 hole nudge

mask (Figure 4.7). Note that the units of these co-ordinates are baseline length divided by observation wavelength (750 nm). In order to correctly sample the PVRs we multiplied the co-ordinates by the wavelength so that co-ordinates are given in meters, then determined the scale of the PVRs in meters.

The original images (Figure 3.6) had a field of view of 150 AU and were 2501 pixels across. Given that the distance to the image was 100 pc, 1 pixel corresponded to $\approx 0.600$ mas. By the Nyquist criterion, the highest frequency in the PVRs will be $\theta \approx 1.200$ mas $\approx 5.815 \times 10^{-9}$ rad. It can be derived from the double slit experiment that $\theta = \dfrac{\lambda}{B}$ where $\lambda$ is the wavelength of observation and $B$ is the baseline length. Hence, the maximum baseline length the PVR can represent is $B = \dfrac{\lambda}{\theta} = \dfrac{750 \text{ nm}}{5.815 \times 10^{-9} \text{ rad}} \approx 129.0$ m. This maximum baseline is the radius of the largest circle that can be drawn inside the PVRs images. The sampling was then done using the `scipy interpolate` module with a 2d cubic interpolation. The sampling of the visibility ratios are shown in Figure 4.10.

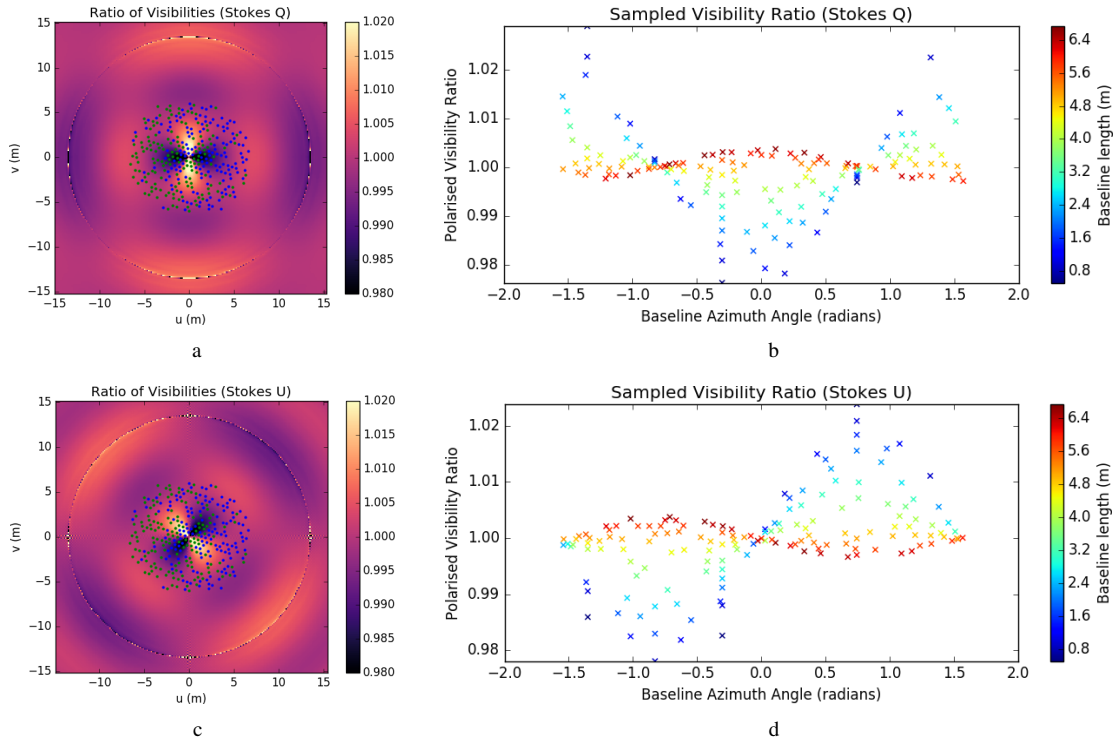



Figure 4.10: Artificial data taken from a model produced by MCFOST (Figure 3.6). Each cross in the sampled visibility ratio corresponds to a point in the ratio of visibilities image.

## 4.1 Reduced $\chi^2$ Error

The $\chi^2$ error is defined as:

$$\chi^2_{\text{reduced}} = \frac{1}{N-p} \sum_{i=0}^{N} \frac{(o_i - m_i)^2}{e_i^2} \tag{1}$$

where $N$ is the number of data points, $p$ is the number of free parameters, $o_i$ is a data point in the observation data, $m_i$ is a data point from the model (Figure 4.10) and $e_i$ is the error in $o_i$. Note that $N - p$ is the degree of freedom.

# 5 The star class

## 5.1 Initialising a star

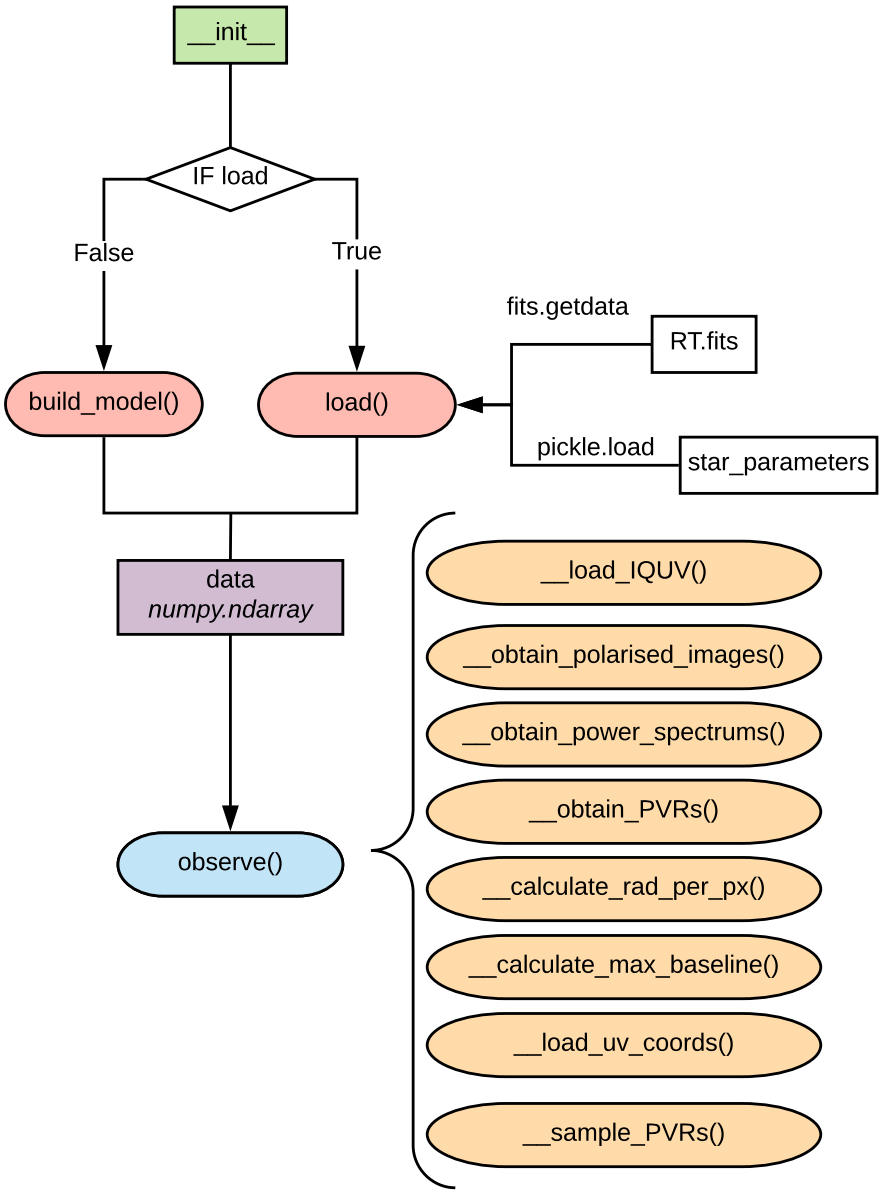



Figure 5.11: Flow chart showing the initialisation of the star class.

## 5.2 Building an MCFOST model

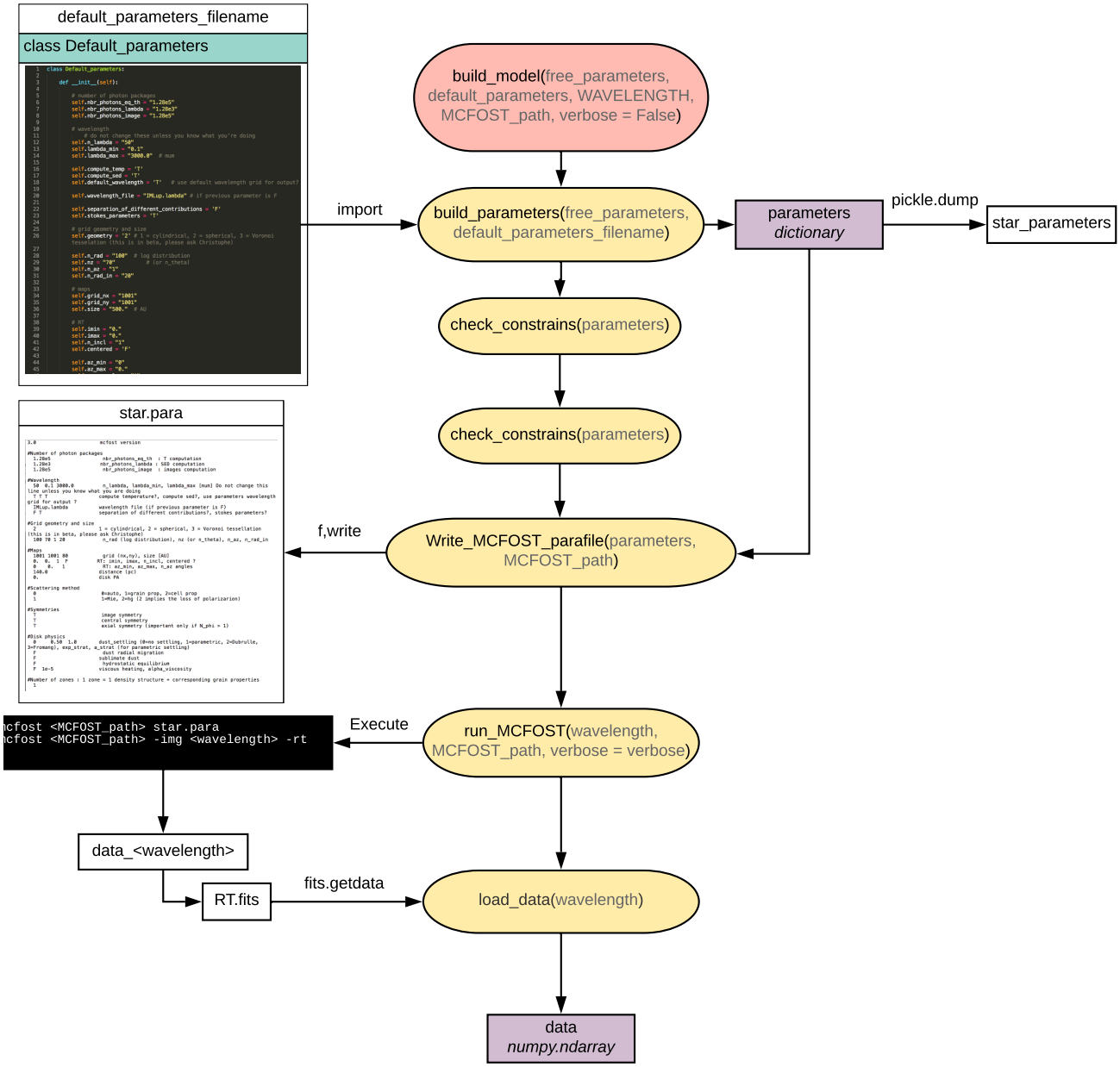



Figure 5.12: Flow chart showing how MCFOST is called to build a model.

## 5.3 Create a simple star

### 5.3.1 Arguments

To create a star, you call the function as follows:

```
Star(free_parameters, default_parameters, WAVELENGTH, MCFOST_path, obs_data)
```

The following is a description of each argument:

| Argument | Description | Type |
|---|---|---|
| free_parameters | A dictionary of free parameters. The name of these parameters must match the naming of these parameters in the default_parameters. | dictionary |
| default_parameters | The name of the default_parameters file. The default parameter file must create an object with the class name `Default_parameters` which has attributes which correspond to the MCFOST parameters. An example default_parameters file is included in Appendix A | String |
| WAVELENGTH | The wavelength of observation in microns. | float |
| MCFOST_path | Path specifying the location where the parameter file will be built and where the data folders will be created | string |
| obs_data | An object containing observation data. See Section 5.3.2 | An object |

Figure 5.13

### 5.3.2 Observation data

Observation data must be stored in an object and passed into the star class. The observation data *must* have the following attributes:

| Attribute | Description | Type |
|---|---|---|
| u_coords, v_coords | The u-v coordinates of each baseline (in instrument coordinates) | np.ndarray |
| blengths | The baseline lengths (in meters) | np.ndarray |
| bazims | The baseline azimuths (in instrument coordinates) | np.ndarray |

Figure 5.14

The attributes above contain information about the mask used during the observation. They are necessary for simulate observations of the model star because they dictate where the polarised visibility ratio is sampled.

If the model star is to be compared with observation data, it is useful to pass the star the following observation data:

| Attribute | Description | Type |
|---|---|---|
| vhvv, vhvvu | The polarised differential visibilities in Stokes Q and U | np.ndarray |
| vhvverr, vhvvuerr | The errors on the polarised differential visibilities in Stokes Q and U | np.ndarray |

Figure 5.15

The observation data is neccessary for some methods (e.g `calculate_reduced_chi2error()`).

Any additional attributes may be helpful, but are not necessary.

### 5.3.3 Demo star

The following is code to create a simple star:

```
import numpy as np
from star import *
from scipy import io

class VAMPIRES_data:

        def __init__(self, filename, cube_info_filename):
                ''' Reads in the polarised-differential calibrated VAMPIRES data produced by the
                  ↪  IDL pipeline:param filename: The full filename of the data file. E.g.
                  ↪  diffdata_vega_.......idlvar '''

                data_object = io.readsav(filename, python_dict=False, verbose=False)
                self.vhvv = data_object.vhvv
                self.vhvverr = data_object.vhvverr
                self.vhvvu = data_object.vhvvu
                self.vhvvuerr = data_object.vhvvuerr
                self.blengths = data_object.blengths
                self.bazims = data_object.bazims
                self.u_coords = data_object.u_coords
                self.v_coords = data_object.v_coords


VAMP_data = "diffdata_RLeo_03_20170313_750-50_18holeNudged_0_0.idlvar"
VAMP_data_info = "cubeinfoMar2017.idlvar"
obs_data = VAMPIRES_data(VAMP_data, VAMP_data_info)

free_params = {"Rin": 3, "Rout": 5, "dust_mass": 1e-7, "radius": 100, "size": 80}
default_params = "default_parameters_star"

WAVELENGTH = 0.75        # microns

# location where the parameter file will be built and where the data folders will be created
MCFOST_path = ""

# Create a star
s = Star(free_params, default_params, WAVELENGTH, MCFOST_path, obs_data, load = False)
```

## 5.4 Optional arguments

These are the optional arguments you can use when creating your star:

| Argument | Description | possible values |
|---|---|---|
| load | Will load in the last RT.fits file produced by MCFOST at the location specified by `MCFOST_path`. The parameters it loads will be those passed to the star on creation, and not necessarily the parameters used to produce the RT.fits file | `True`, `False`, default: `False` |
| verbose | Will either display or suppress all the MCFOST. calculations shown when running MCFOST | `True`, `False`, default: `False` |

## 5.5 Attributes

After initialisation the star will have the following attributes:

| Attribute | Description | Type |
|---|---|---|
| WAVELENGTH | The wavelength of observation in microns | float |
| MCFOST_path | Path specifying the location where the parameter file will be built and where the data folders will be created | string |
| free_parameters | A dictionary carrying all the free parameters that will overwrite the default parameters. The keys are the parameter names (must match the parameter names in default_parameters) and the values are strings | dictionary |
| default_parameters | The name of the python file defining a `Default_parameter` class. | string |
| n_free_parameters | The number of free parameters. i.e. the number of keys in free_parameters | int |

| | | |
|---|---|---|
| parameters | A `Default_parameters` object which is defined in the default_parameters python file. The attributes of the object correspond to the MCOST parameters | Default_parameters object |
| obs_data | An object which holds all the observational data. See Section 5.3.2 | object |
| resolution | The resolution of the image in radians. Tells you the smallest thing in the IQUV images that we can detect | numpy.float64 |
| resolution_mask_soft | The resolution of the mask in mas for object with a soft edge. $R = \lambda/B$ | numpy.float64 |
| resolution_mask_hard | The resolution of the mask in mas for an object with a hard edge. $R = 1.22\lambda/B$ | numpy.float64 |
| max_baseline | The maximum baseline that can be represented in the PVRs | numpy.float64 |
| angular_size_image_rad | Gives the angular size of the image in radians | numpy.float64 |
| rad_per_px | The angular size of a pixel in radians | numpy.float64 |
| mas_per_px | The angular size of a pixel in mas | numpy.float64 |
| distance_m | Distance to the star in meters | float |
| distance_AU | Distance to the star in AU | float |
| radius_m | Radius of the star in meters | float |
| diameter_mas | Diameter of the star in mas | numpy.float64 |
| rin_m | The inner radius of the dust shell in meters | float |
| din_mas | The inner diameter of the dust shell in mas | numpy.float64 |
| rout_m | The outer radius of the dust shell in meters | float |
| dout_mas | The outer diameter of the dust shell in mas | numpy.float64 |
| I, Q, U, V | Images produced by MCFOST corresponding to the polarised images in Stokes I, Q, U and V | numpy.ndarray |
| Hp, Vp, Ap, Bp | Polarised images defined by: $Hp = I + Q, Vp = I - Q, Ap = I + U, Bp = I - U$ | numpy.ndarray |
| Hps, Vps, Aps, Bps | The power spectrums of Hps, Vps, Aps and Bps where the power spectrum of $\vec{x}$ is given by $FFT2(\|\vec{x}\|^2)$. The power spectrums have been recentered so that the maximum value is at the center of the image, and normalised by dividing through the image by the maximum value | numpy.ndarray |
| PVR_Q, PVR_U | The polarised visibility ratio. PVR_Q = Hps / Vps, PVR_U = Aps / Vps | numpy.ndarray |
| U_vals, V_vals | The grid spacings of the PVR images in terms of baseline length in meters. The edges of the images correspond to $\pm$max_baseline and the center is at 0 meters | numpy.ndarray |
| u_meters, v_meters | Lists storing the u, v co-ordinates of the baselines in meters. The co-ordinates of baseline$_i$ is stored at index $i$ in both lists | list |
| u_neg_meters, v_neg_meters | The same as u_meters and v_meters but each value is multiplied by $-1$ | list |
| data_Q, data_U | Stores the polarised visibility ratios obtained from sampling PVR_Q and PVR_U. The order of the samples are in the same order as the baselines i.e. sample$_i$ corresponds to baseline$_i$ | numpy.ndarray |
| data | The data from the fits file produced by MCFOST unpacked using `fits.getdata` | numpy.ndarray |
| *reduced_chi2err_Q | The reduced $\chi^2$ error of data_Q compared with obs_data.vhvv | numpy.float64 |
| *reduced_chi2err_U | The reduced $\chi^2$ error of data_U compared with obs_data.vhvvu | numpy.float64 |
| *reduced_chi2err | The average of reduced_chi2err_Q and reduced_chi2err_U | numpy.float64 |
| †data_I | Values from sampling Ips the same way PVR_Q and PVR_U are sampled to obtain data_Q and data_U | numpy.ndarray |

Figure 5.16: Attributes of the Star class.
* Attribute not created during the initialisation process. Requires `calculate_reduced_chi2err()` to be called first.
† Attribute not created during the initialisation process. It is only calculated in `sanity_check_star()`

## 5.6 Methods

### 5.6.1 Callable methods

| Method | Description | Optional arguments and requirements |
|---|---|---|
| observe() | Simulates the observation of the star. It produces interferometric data: the polarised differential visibilities in Stokes Q and U. Executed upon initialisation. | None |
| display_IQUV() | Displays the MCFOST images corresponding to the polarised images in Stokes I, Q U and V. | None |
| display_power_spectrums() | Displays the power spectrums of H, V, A and B | None |
| display_PVRs() | Displays the polarised visibility ratios | None |
| display_data() | Provides an easy way to plot the simulated and observed data. More details in Section 5.6.6 | Some options require observation data outlined in Table 5.15 |
| print_parameters() | Prints the MCFOST parameters and their values in alphabetical order | None |
| calculate_reduced_chi2error() | Calculated the reduced $\chi^2$ error (defined by Equation 1. Returns the reduced $\chi^2$ error in both Stokes Q and U, as well as overall.) | Observational data specified in Table 5.15 |
| checksum() | For monochromatic coherent radiation, $I^2 = Q^2 + U^2 + V^2$. This method returns $I^2 - Q^2 - U^2 - V^2$. It is expected that this result is close to 0. | None |
| sanity_check_star() | A basic check looking at a star with no shell. See Section 6.1 | None |
| sanity_check_shell() | A basic check looking at a just resolved star with a shell twice the size of the star. See section 6.2 | None |

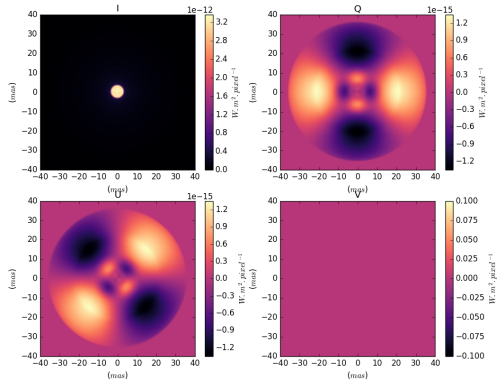
### 5.6.2 Hidden methods

| Method | Description |
|---|---|
| __load_IQUV() | Unloads the fits file into the IQUV images |
| __obtain_polarised_images() | Obtains the polarised images (Hp, Vp, Ap, Bp) from the IQUV images |
| __obtain_power_spectrums() | Obtains the power spectrums (Hps, Vps, Aps, Bps) from the Hp, Vp, Ap, Bp images |
| __obtain_PVRs() | Obtains the polarised visibility ratios in Stokes Q and U (PVR_Q, PVR_U) from the power spectrums Hps, Vps, Aps, Bps |
| __calculate_rad_per_px() | Calculates the angular size of a pixel in radians |
| __calculate_max_baseline() | Determines the maximum baseline that the PVR_Q and PVR_U images can represent and is used to determine where to sample the PVRs |
| __load_uv_coords() | Determines the uv co-ordinates where we should sample the PVRs in units of meters |
| __sample_PVR() | A function called by __sample_PVRs(). This function samples a single PVR using the `interpolate` module with a 2d quintic interpolation. |
| __sample_PVRs() | Samples PVR_Q and PVR_U. The samples are stored in data_Q and data_U |
| __display_sim_data() | Plots the simulated data of either Stokes Q or Stokes U on a specified ax (plot) |
| __display_observed_data() | Plots the observed data of either Stokes Q or Stokes U on a specified ax (plot) |
| __sampled_power_spectrum() | This samples the power spectrum of I (Ips) the same way sample_PVR() samples PVR_Q and PVR_U |
| __display_sampled_power_spectrum_I() | Plots Ips with the sample points overlaid on the image and data_I as a function of baseline length. |

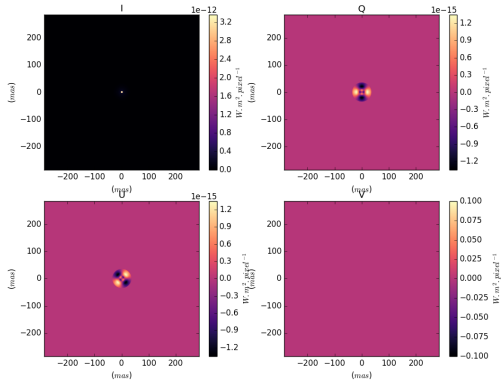
### 5.6.3 `display_IQUV(scale = "auto", interpolation = None)`

By default, `scale = "auto"` automatically zooms into the images to show the outer radius of the shell with a 10% margin. To turn this off set `scale = "off"`. Note: This does not work if the outer edge of the envelope extends beyond the figure.
By default, `interpolation = None` and is passed as an argument into `imshow()`. To turn interpolation off, set `interpolation = "nearest"`.

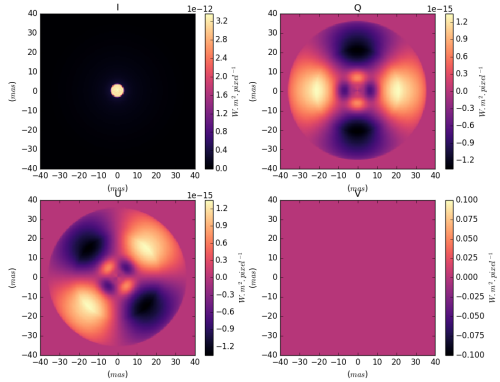
```
display_IQUV()
plot.show()
```



```
display_IQUV(scale = "off")
plot.show()
```



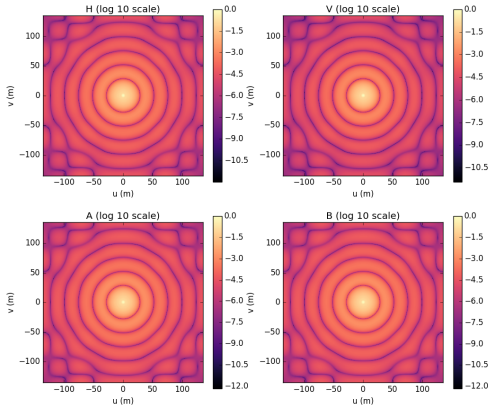
```
display_IQUV(interpolation = "nearest")
plot.show()
```



### 5.6.4 `display_power_spectrums()`

Displays the power spectrums (Hps, Vps, Aps, Bps) on a log 10 scale.

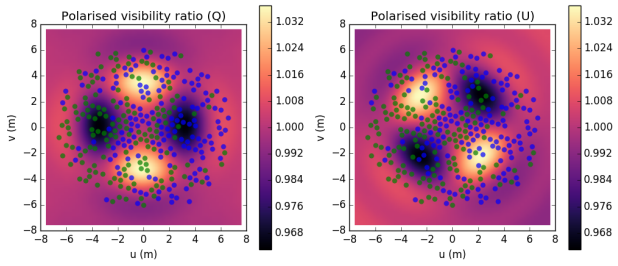
```
display_power_spectrums()
plot.show()
```



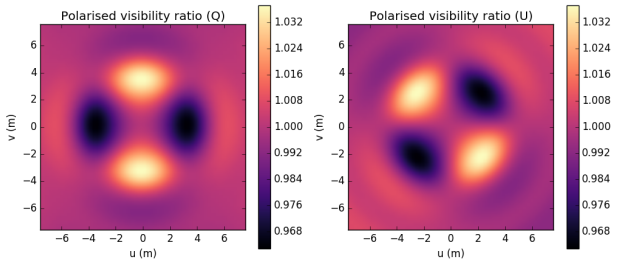
### 5.6.5 `display_PVRs(overplot = True, scale = "auto", interpolation = None, color_scale = "auto")`

Displays the polarised visibility ratios (PVR_Q, PVR_U). It automatically plots the points where the PVRs are sampled. To turn this off, set `overplot = False`. The images automatically scale to display the region of the PVRs where they are sampled. To view the entire PVR set `scale = "off"`. The images automatically adjust the colour scale so that the maximum and minimum colour values correspond to the maximum value of the sampled points, with the colourbar centered at 1. To manually adjust the colour bar so that it ranges from $1 - \alpha$ to $1 + \alpha$, set `color_scale = `$\alpha$.

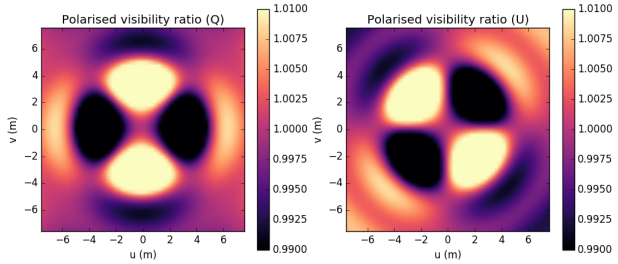
```
display_PVRs()
plot.show()
```



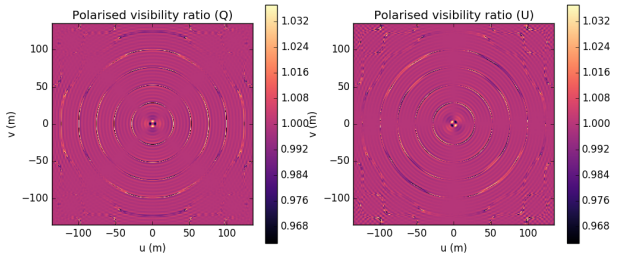

```
display_PVRs(overplot = False)
plot.show()
```




```
display_PVRs(overplot = False, color_scale = 0.01)
plot.show()
```




```
display_PVRs(overplot = False, scale = "off")
plot.show()
```




### 5.6.6 `display_data(option = 6, ylims = "auto", epsilon = 0.001)`

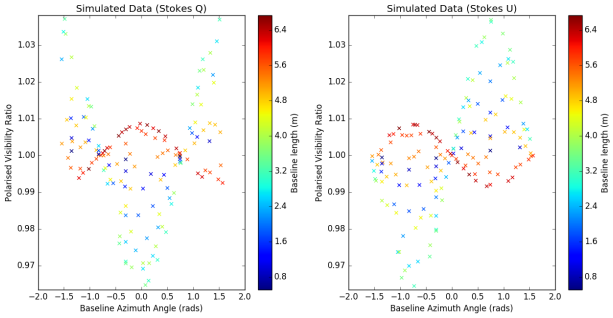
For these images, we have subtracted out the bias from the observation data using the following lines:

```
    self.vhvv -= np.mean(self.vhvv) - 1
    self.vhvvu -= np.mean(self.vhvvu) - 1
```

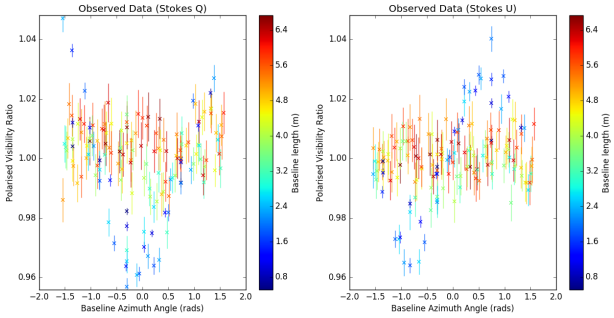
The `display_data()` function has 6 options that displays various combinations of the simulated and observed data:

- **Option 1**: simulated data
- **Option 2**: observed data
- **Option 3**: simulated and observed data in Stokes Q
- **Option 4**: simulated and observed data in Stokes U
- **Option 5**: simulated and observed data in Stokes Q and U (separate plots)
- **Option 6 (default)**: simulated and observed data in Stokes Q and U (Q's and U's on the same plot)

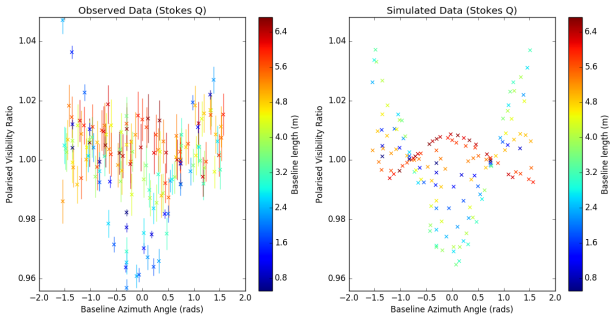
```
display_data(option = 1)
plot.show()
```



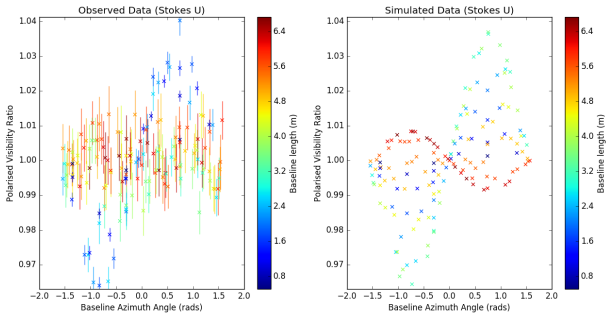

```
display_data(option = 2)
plot.show()
```



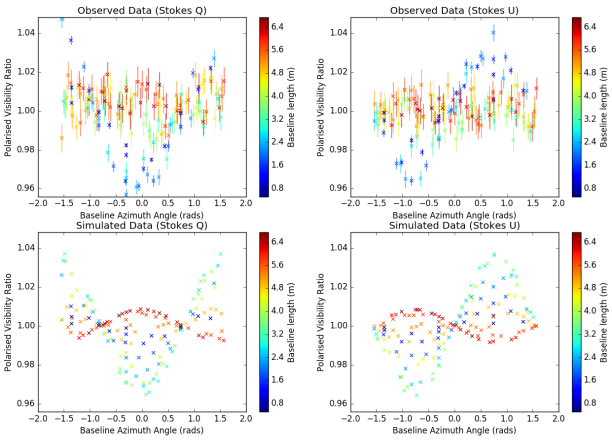

```
display_data(option = 3)
plot.show()
```



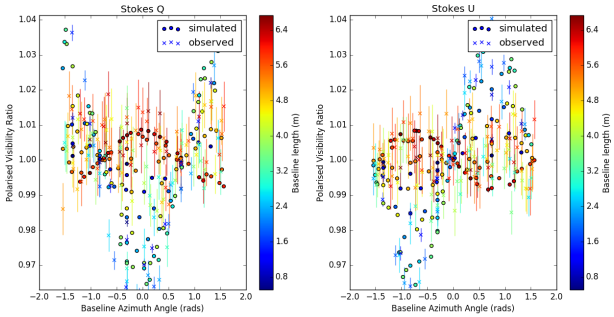

```
display_data(option = 4)
plot.show()
```




```
display_data(option = 5)
plot.show()
```

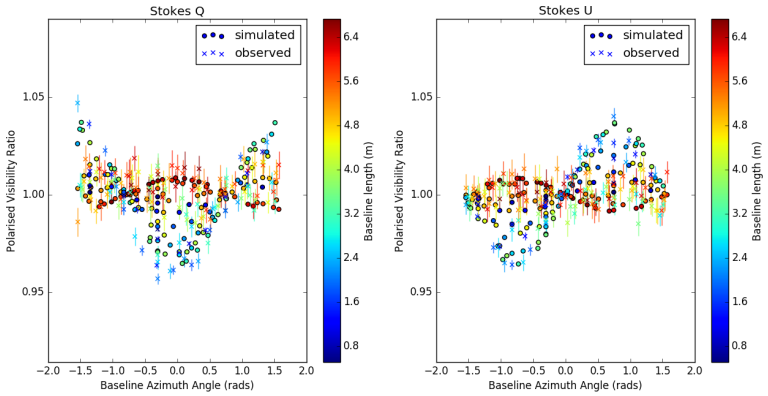



```
display_data()
plot.show()
```



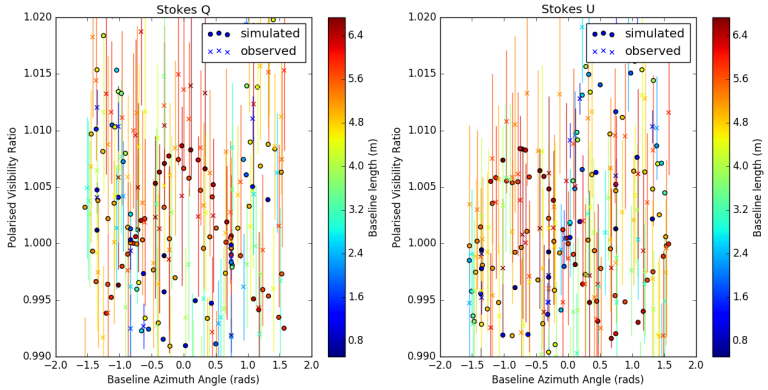

The y limits are automatically adjusted so that the maximum y value is $max + \epsilon$ and the minimum y value is $min - \epsilon$ where $max$ and $min$ are the maximum and minimum values of the data points being desplayed. $\epsilon$ can be changed by specifying `epsilon`.

```
display_data(epsilon = 0.05)
plot.show()
```




Alternatively, you can specify the values of the ylims by passing `ylims` a list with the minimum and maximum y values.

```
display_data(ylims = [0.99, 1.02])
plot.show()
```



# 6 Sanity checks

## 6.1 Sanity check 1: Star

### 6.1.1 Worked example

Here you can see a working example of the first sanity check. The default parameters were specified in appendix A. The free parameters used are:

```
free_params = {"Rin": 3, "Rout": 5, "dust_mass": 1e-20, "radius": 500, "size":
↪  80}
```

and the sanity check was called with:

```
s.sanity_check_star()
```

and the following images were produced:

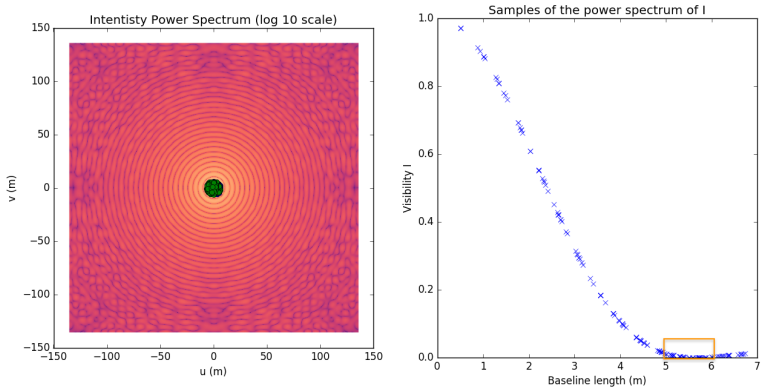



Figure 6.17: A figure produced during the star sanity check.

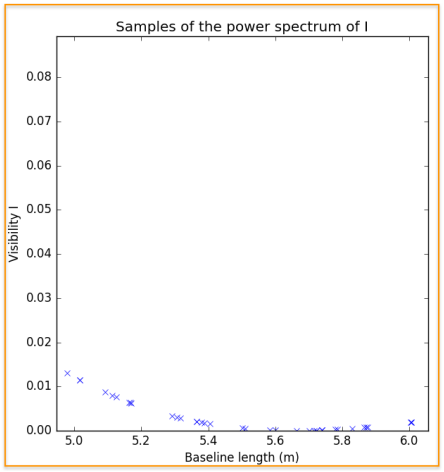



Figure 6.18: Figure 6.17 zoomed in on the orange rectangle. The visibility goes to 0 between 5.4 and 5.8 meters.

As can be seen in Figure 6.18, the visibility goes to 0 between 5.4 and 5.8 meters. Enter these values into the terminal where prompted.

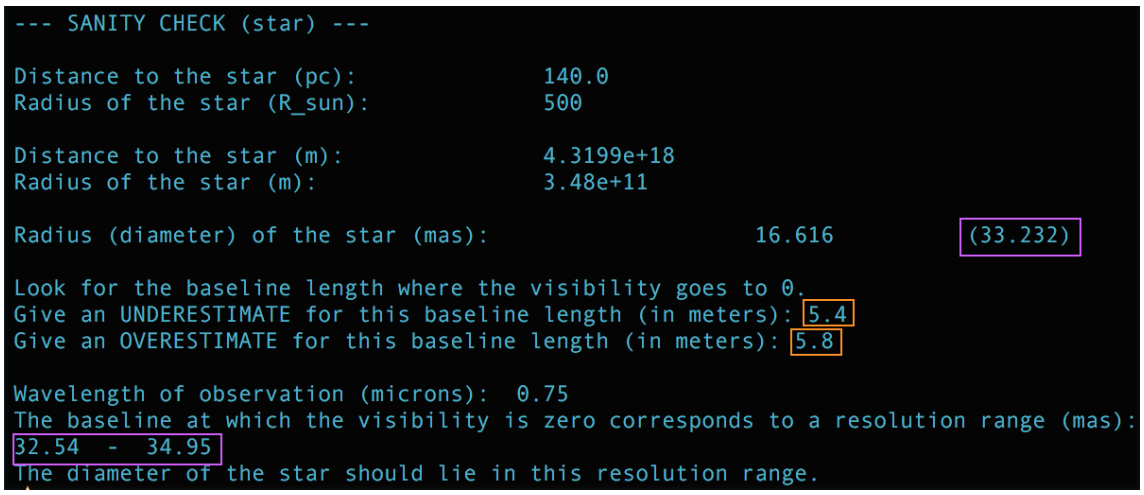



Figure 6.19: Terminal output for the sanity check. Enter the range of baselines where the visibility goes to 0 as shown in Figure 6.18. The terminal should estimate a range for the diameter of the star which should match the actual diameter of the star.

The terminal should then give you an estimate for the diameter of the star. In Figure 6.19 this range is 32.54 - 34.95 mas. We can see that the diameter of our star is 33.232 which does indeed lie in the provided range.

### 6.1.2 Optional Arguments

`sanity_check_star(self, star_edge = "hard", shell = False, prec = 5)`

| | |
|---|---|
| `star_edge` | The determines how resolution ($R$) is calculated for a given wavelength ($\lambda$) and baseline ($B$). If the edge is hard, $R = 1.22\lambda/B$, if the edge is soft, $R = \lambda/B$. |
| `shell` | If `shell = True`, the terminal output will display the inner and outer radius of the dust shell in AU, meter and mas. |
| `prec` | Cahnges the precision of the values that are displayed in the terminal. |

### 6.1.3 Trouble shooting: too much dust (the predicted diameter of the star is wrong)

If you try to run this sanity check on a star that has a substantial dust shell, it won't work!
Significantly reduce the dust mass!

The following demonstrates what happens when you run this sanity check with too much dust.

Performing a sanity check with the following parameters:

```
free_params = {"Rin": 3, "Rout": 5, "dust_mass": 1e-6, "radius": 500, "size":
 ↪  80}
```

The following figure is produced:

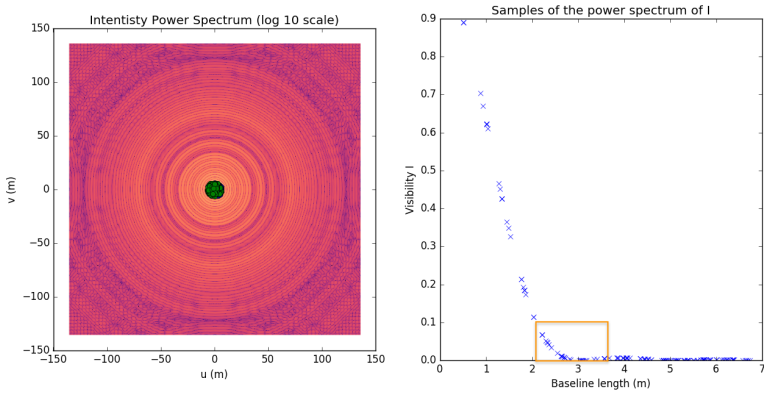



Figure 6.20: An example of a figure produced during a star sanity check of a star with a dust shell.

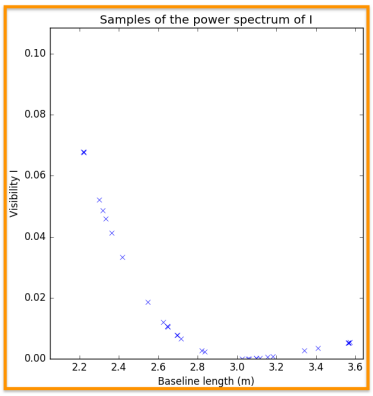



Figure 6.21: Figure 6.20 zoomed in on the orange rectangle.

Reading off Figure 6.21, the baseline at which the visibility of I goes to 0 is between **2.8** and **3.2** meters. Feeding this as input into the terminal produces the following output:

```
--- SANITY CHECK (star) ---

Distance to the star (pc):              140.0
Radius of the star (R_sun):             500

Distance to the star (m):               4.3199e+18
Radius of the star (m):                 3.48e+11

Radius (diameter) of the star (mas):                    16.616          (33.232)

Look for the baseline length where the visibility goes to 0.
Give an UNDERESTIMATE for this baseline length (in meters): 2.8
Give an OVERESTIMATE for this baseline length (in meters): 3.2

Wavelength of observation (microns):  0.75
The baseline at which the visibility is zero corresponds to a resolution range (mas):
58.979  -  67.404
The diameter of the star should lie in this resolution range.
```

Figure 6.22: Terminal output after providing an underestimate of 2.8 and an overestimate of 3.2 meters of where the visibility of I goes to 0. This corresponds to a resolution range of 58.979 - 67.404 mas which is inconsistent with the daimeter of the star (33.232 mas).

### 6.1.4 Trouble shooting: unresolved star (visibility doesn't reach zero)

Check the resolution (mas) of the mask with:

```
print(s.resolution_mask_soft)
```

If your star is smaller than the resolution of the mask than this sanity check will not work!
Make your star bigger!

Here is an example of an unresolved star:

```
free_params = {"Rin": 3, "Rout": 5, "dust_mass": 1e-20, "radius": 300, "size":
↪  80}
```

The size of the star is mas is displayed in the terminal output.

```
--- SANITY CHECK (star) ---

Distance to the star (pc):              140.0
Radius of the star (R_sun):             300

Distance to the star (m):               4.3199e+18
Radius of the star (m):                 2.088e+11

Radius (diameter) of the star (mas):                    9.9696          (19.939)

Look for the baseline length where the visibility goes to 0.
Give an UNDERESTIMATE for this baseline length (in meters):
```

Figure 6.23: Terminal output for a star that is unresolved. The diamter of the star is 19.939 mas and the resolution of the mask is 28 mas.

The figures produces by an unresolved star might look like this:

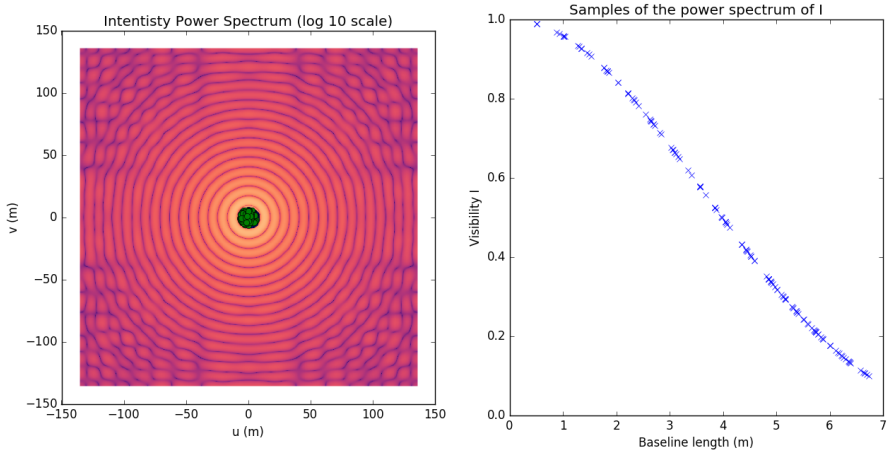



Figure 6.24: Figures from an unresolved star. The visibility never reaches zero.

The visibility of I for an unresolved will never reach 0. If it appears to reach the x-axis, check the scale on the y-axis, because the figure might automatically scale.

## 6.2 Sanity check 2: Shell

### 6.2.1 Worked example

Here you can see a working example of the second sanity check. The default parameters for the star were as specified in appendix A. A star was created without modifying the default parameters and at a wavelength of 0.75 microns. The *shell* sanity check was called.

```
free_params = {}
```

and the sanity check was called with:

```
s.sanity_check_star()
```

A window with a figure should pop up. This can be closed and ignored for now. The terminal output should look like Figure. 6.25

```
--- SANITY CHECK (shell) ---

The maximum baseline for this mask is (meters)  6.7378
The resolution of this mask is (mas):           28.011
Distance to the star (pc):                      140.0

A star that is just resolved has radius (solar radii):          421.45
A shell that is twice the size of the star has radius (AU):     3.9216

--- Make a star that is just resolved with a shell twice the size of the star ---
Radius of the star (R_sun):             200.0
Inner radius of dust shell (AU):        3.0
Outer radius of dust shell (AU):        300.

Distance to the star (m):       4.3199e+18
Radius of the star (m):         1.392e+11
Inner radius of dust shell (m): 4.4879e+11
Outer radius of dust shell (m): 4.4879e+13

Radius (diameter) of the star (mas):                    6.6464  (13.293)
Inner radius (diameter) of the dust shell (mas):        21.429  (42.857)
Outer radius (diamter) of the dust shell (mas):         2142.9  (4285.7)

You should see distinct sinusoidal shapes. If you do not, you may need to zoom in.
Look at the blue-green sinusoid where the amplitude is largest. The colour tells you baseline.
What is an UNDERESTIMATE for the baseline length of this sinusoid?
```

Figure 6.25: Terminal output after running `sanity_check_shell()`. The box indicates the lines that tell: the radius of a star (in solar radii) that is just resolved at the given distance with the given mask, and the corresponding radius of a dust shell (in AU) twice the radius of the star.

The terminal will be waiting for input, but for now, it is not important, so you can terminate the program.
You will see a box Figure 6.25. The box indicates the lines that tell: the radius of a star (in solar radii) that is just resolved at the given distance with the given mask, and the corresponding radius of a dust shell (in AU) twice the radius of the star.
We created a star with these parameters. We wanted to make the dust shell thin, so we made the inner radius (`Rin`) and outer radius (`Rout`) approximately the same. We used the following parameters:

```
free_params = {"Rin": 4, "Rout": 4.1, "dust_mass": 1e-10, "radius": 422,
 ↪  "size": 101}
```

You may need to fiddle with the amount of dust. You need enough dust to obtain a polarised signal, but if you have too much dust you will be unable to 'see through' the dust shell.
The second half of the terminal output should change, as shown if Figure 6.26.

```
--- SANITY CHECK (shell) ---

The maximum baseline for this mask is (meters)  6.7378
The resolution of this mask is (mas):           28.011
Distance to the star (pc):                      140.0

A star that is just resolved has radius (solar radii):          421.45
A shell that is twice the size of the star has radius (AU):     3.9216

--- Make a star that is just resolved with a shell twice the size of the star ---
Radius of the star (R_sun):             422
Inner radius of dust shell (AU):        4
Outer radius of dust shell (AU):        4.1

Distance to the star (m):       4.3199e+18
Radius of the star (m):         2.9371e+11
Inner radius of dust shell (m): 5.9839e+11
Outer radius of dust shell (m): 6.1335e+11

Radius (diameter) of the star (mas):                    14.024  (28.048)
Inner radius (diameter) of the dust shell (mas):        28.571  (57.143)
Outer radius (diamter) of the dust shell (mas):         29.286  (58.571)

You should see distinct sinusoidal shapes. If you do not, you may need to zoom in.
Look at the blue-green sinusoid where the amplitude is largest. The colour tells you baseline.
What is an UNDERESTIMATE for the baseline length of this sinusoid?
```

Figure 6.26: Terminal output after changing the free parameters of the star to create a star that is just resolved with a thin shell with radius twice that of the star.

To look at your star, you can add the following line to your code:

```
s.display_IQUV(interpolation = "nearest")
```

```
plot.show()
```

Our star is shown in Figure 6.27.

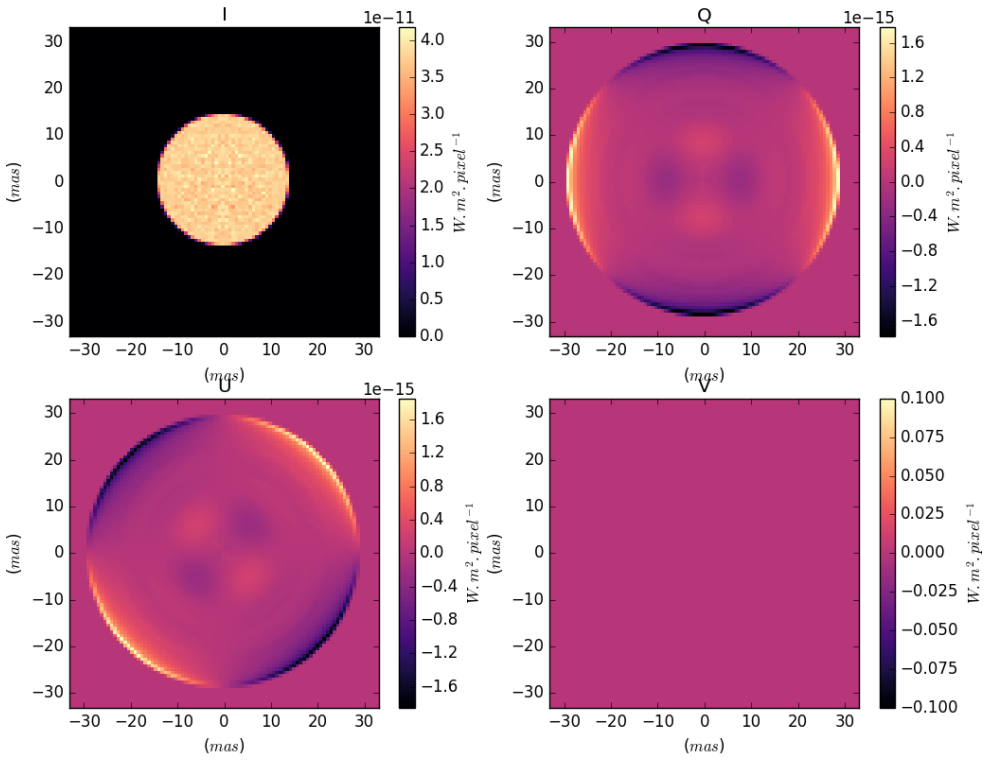



Figure 6.27: Images produced by MCFOST in Stokes polarised intensities.

You should be able to zoom in and check that the radius of the star and the shell are consistent with the input parameters (Figure 6.26).

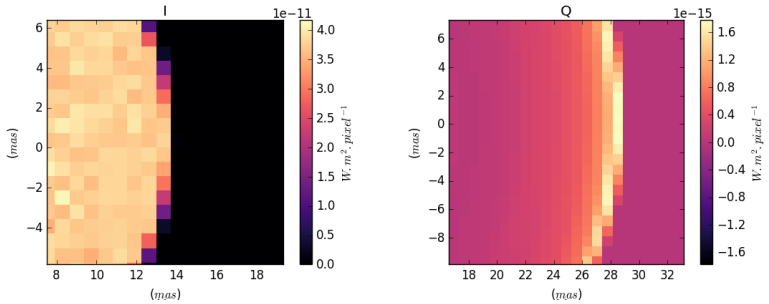



Figure 6.28: I and Q of Figure. 6.27 zoomed in. This is consistent with a stellar radius of 14.0 mas and a shell inner and outer radius of 27.9 mas and 29.1 mas as read off the output shown in Figure 6.26.

A window with another figure should also pop up (Figure 6.29). You may need to zoom in on the data centered around 1.00. Data points much larger or smaller than 1.00 are due to noise and can be ignored for this test.

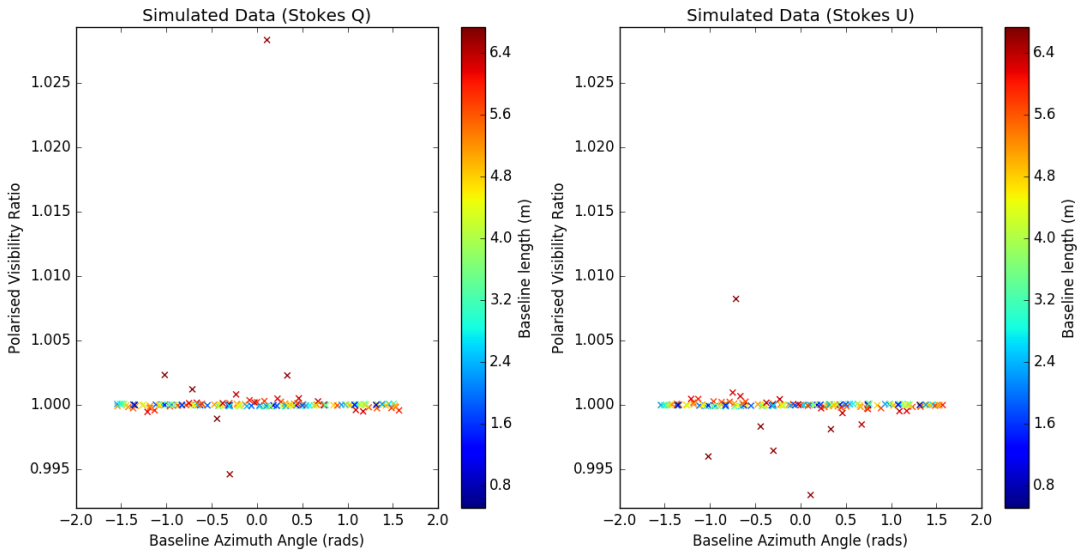



Figure 6.29: Simulated polarimetric differential data produced from the MCFOST images shown in Figure 6.27 by sampling the polarised visibility ratios. The figures automatically scale to show all data points. Noise causes some data points to stray large amounts from 1.00.

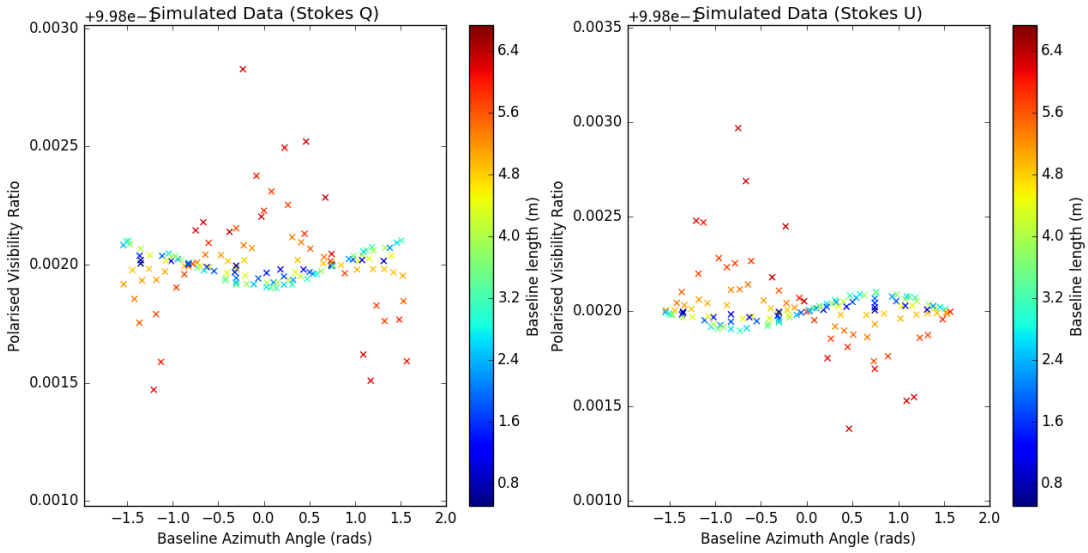



Figure 6.30: The same data as shown in 6.29 but zoomed in around 1.00.

You will notice that there appear to be two sinusoidal functions. One of these is in red. This corresponds to features of the star since the star we created is just resolved so is picked up by the longest baselines (red). The other sinusoid (in bright blue and green) corresponds to features of the shell. We are interested in the baseline that corresponds to this sinusoid with the largest amplitude. By eye, this sinusoid looks to be light blue to green, which corresponds to a range of baselines of 2.8 to 4.0 meters.
Enter these values into the terminal prompt.

```
You should see distinct sinusoidal shapes. If you do not, you may need to zoom in.
Look at the blue-green sinusoid where the amplitude is largest. The colour tells you baseline.
What is an UNDERESTIMATE for the baseline length of this sinusoid? 2.8
What is an OVERESTIMATE for the baseline length of this sinusoid? 4.0
```

Figure 6.31: Input the range of baselines (2.8 - 4.0 meters) that correspond to the light blue-green sinusoid in Figure 6.30.

Directly after this, the terminal should give you a range telling you what the diameter of the shell should be.

```
Wavelength of observation (microns): 0.75
The diameter of the SHELL should lie between (mas):
47.183  -  67.404
```

Figure 6.32: An estimate for the shell diameter calculated from Figure 6.29. It predicts the diameter of the shell should be between 47 and 67 mas.

We see that an estimate of our shell diameter uisng Figure 6.29 is between 47 and 67 mas. If we look at the properties of our star in Figure 6.26, our shell had an inner diameter of 56 mas and an outer diameter of 58 mas, which is consistent with our estimation.

### 6.2.2 Optional Arguments

`sanity_check_shell(self, star_edge = "hard", dp = 5, prec = 5)`

| | |
|---|---|
| `star_edge` | The determines how resolution ($R$) is calculated for a given wavelength ($\lambda$) and baseline ($B$). If the edge is hard, $R = 1.22\lambda/B$, if the edge is soft, $R = \lambda/B$. |
| `prec` | Cahnges the precision of the values that are displayed in the terminal. |

### 6.2.3 Trouble Shooting: too much dust

If your star has too much dust, this check won't work!
Reduce the dust mass!
For example, the following free parameters produce the images show in Figure 6.33. You cannot distinguish the star from the shell in these images.

```
free_params = {"Rin": 4, "Rout": 4.1, "dust_mass": 1e-7, "radius": 422, "size":
 ↪  101}
```

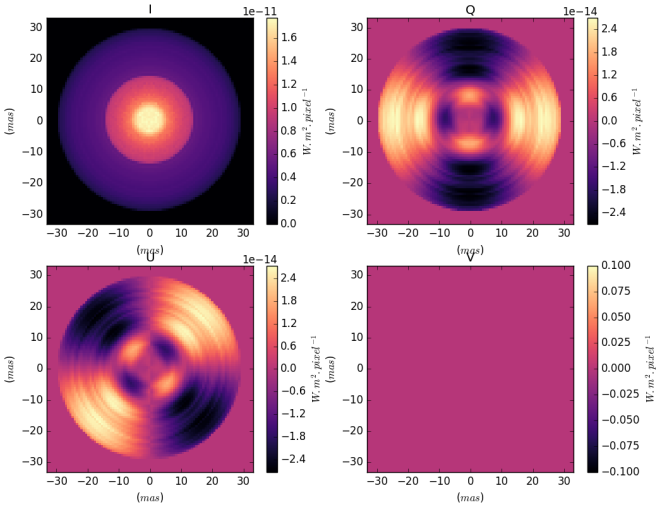



Figure 6.33: A star with too must dust. The shell sanity check will not work.

The following is the simulated data corresponding to these images:

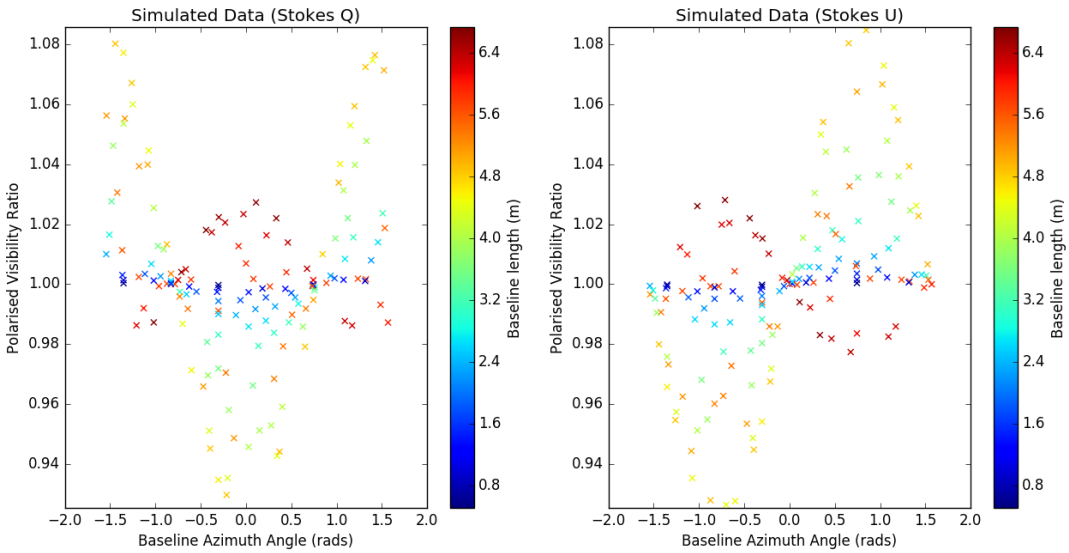



Figure 6.34: Simulated data from the images shown in figure 6.33

You can see that the sinusoid is now yellow-orange colour instead of light blue-green. This correspons to a baseline range of 4.0-4.4 which estimates the diameter of the shell to be between 42.894 and 47.183 mas which is an *underestimate* for the diameter of the dust shell which is 58 mas.

# A Default parameter class

```
class Default_parameters:

        def __init__(self):

                # number of photon packages
                self.nbr_photons_eq_th = "1.28e5"
                self.nbr_photons_lambda = "1.28e3"
                self.nbr_photons_image = "1.28e5"

                # wavelength
                        # do not change these unless you know what you're doing
                self.n_lambda = "50"
                self.lambda_min = "0.1"
                self.lambda_max = "3000.0"        # mum

                self.compute_temp = 'T'
                self.compute_sed = 'T'
                self.default_wavelength = 'T'         # use default wavelength grid for output?

                self.wavelength_file = "IMLup.lambda" # if previous parameter is F

                self.separation_of_different_contributions = 'F'
                self.stokes_parameters = 'T'

                # grid geometry and size
                self.geometry = '2' # 1 = cylindrical, 2 = spherical, 3 = Voronoi tesselation (this
                 ↪  is in beta, please ask Christophe)

                self.n_rad = "100"        # log distribution
                self.nz = "70"                  # (or n_theta)
                self.n_az = "1"
                self.n_rad_in = "20"

                # maps
                self.grid_nx = "1001"
                self.grid_ny = "1001"
                self.size = "500."        # AU

                # RT
                self.imin = "0."
                self.imax = "0."
                self.n_incl = "1"
                self.centered = 'F'

                self.az_min = "0"
                self.az_max = "0."
                self.n_az_angles = "1"

                self.distance = "140.0"        # pc
                self.disk_PA = "0."

                # Scattering method
                self.scattering_method = "0"         # 0=auto, 1=grain prop, 2=cell prop
                self.mie_hg = "1"                               # 1=Mie, 2=hg (2 implies the loss
                 ↪  of polarizarion)

                # Symmetries
                self.image_symmetry = 'T'
                self.central_symmetry = 'T'
                self.axial_symmetry = 'T'         # (important only if N_phi > 1)

                # Disk physics
                self.dust_settling = "0"        # (0=no settling, 1=parametric, 2=Dubrulle,
                 ↪  3=Fromang)
                self.exp_strat = "0.50"
                self.a_strat = "1.0"                # (for parametric settling)

                self.dust_radial_migration = 'F'
                self.sublimate_dust = 'F'
                self.hydrostatic_equilibrium = 'F'

                self.viscous_heating = 'F'
                self.alpha_viscosity = "1e-5"

```

```
        # Number of zones: 1 zone = 1 density structure + corresponding grain properties
        self.number_of_zones = "1"

        # Density structure
        self.zone_type = "3"        # zone type : 1 = disk, 2 = tappered-edge disk, 3 =
         ↪  envelope, 4 = debris disk, 5 = wall

        self.dust_mass = "1.e-6"
        self.gas_to_dust_mass_ratio = "100."

        self.scale_height = "10."
        self.reference_radius = "100.0"        # AU, unused for envelope
        self.vertical_profile_exponent = "2"        # only for debris disk

        self.Rin = "3.0"
        self.edge = "0.0"
        self.Rout = "300."
        self.Rc = "100."                # AU, Rc is only used for tappered-edge & debris
         ↪  disks (Rout set to 8*Rc if Rout==0)

        self.flaring_exponent = "1.125"        # unused for envelope

        self.surface_density_exponent = "-0.5"        # (or -gamma for tappered-edge disk or
         ↪  volume density for envelope), usually < 0
        self.negative_gamma_exp = "0.0"        # or alpha_in & alpha_out for debris disk

        # Grain properties
        self.number_of_species = "1"

        self.grain_type = "Mie"        # Mie of DHS
        self.N_components = "1"
        self.mixing_rule = "2" # 1 = EMT or 2 = coating
        self.porosity = "0.0"
        self.max_fraction = "1.0"
        self.Vmax = "0.9"        # for DHS

        self.optical_indicies_file = "Draine_Si_sUV.dat"
        self.volume_fraction = "1.0"

        self.heating_method = "1"        # 1 = RE + LTE, 2 = RE + NLTE, 3 = NRE

        self.amin = "0.03"        # mum
        self.amax = "1000.0" # mum
        self.aexp = "3.5"
        self.n_grains = "100"        # log distribution

        # Molecular RT settings
        self.lpop = 'T'
        self.laccurate_pop = 'T'
        self.LTE = 'T'
        self.profile_width = "15."        # km.s^-1

        self.v_turb = "0.2"        # delta

        self.nmol = "1"

        self.molecular_data_filename = "co@xpol.dat"
        self.level_max = "6"

        self.vmax = "1.0"        # km.s^-1
        self.n_speed = "20"

        self.cst_molecule_abundance = 'T'
        self.abundance = "1.e-6"
        self.abundance_file = "abundance.fits.gz"

        self.ray_tracing = 'T'
        self.number_lines_in_RT = "3"

        self.transition_number_1 = "1"
        self.transition_number_2 = "2"
        self.transition_number_3 = "3"

        # Star properties
        self.number_of_stars = "1"
        self.temp = "4000.0"
```

```
145                 self.radius = "200.0"         # solar radius
146                 self.mass = "1.0"          # solar mass
147                 self.x = "0.0"          # AU
148                 self.y = "0.0"          # AU
149                 self.z = "0.0"          # AU
150                 self.is_blackbody = 'T'
151                 self.fUV = "0.1"
152                 self.slope_FUV = "2.2"
```

# References


W. Singer, M. Totzeck, and H. Gross, *Handbook of Optical Systems, Volume 2, Physical Image Formation*, Oct. 2005.

A. Labeyrie, S. Lipson, and P. Nisenson, *An Introduction to Optical Stellar Interferometry*. Cambridge University Press, 2006. [Online]. Available: https://books.google.com.au/books?id=l3cSxcRmnu8C

H. Fizeau, "Prix Borodin: Rapport sur le concours de l'année 1867," *Comptes Rendus de l'Académie des Sciences*, vol. 66, p. 932, 1868.

A. A. Michelson, "Measurement of Jupiter's Satellites by Interference," *Nature*, vol. 45, pp. 160–161, Dec. 1891.

A. A. Michelson and F. G. Pease, "Measurement of the Diameter of $\alpha$ Orionis with the Interferometer." *ApJ*, vol. 53, May 1921.

T. Young, "The Bakerian Lecture: On the Theory of Light and Colours," *Philosophical Transactions of the Royal Society of London Series I*, vol. 92, pp. 12–48, 1802.

P. G. Tuthill, J. D. Monnier, W. C. Danchi, E. H. Wishnow, and C. A. Haniff, "Michelson Interferometry with the Keck I Telescope," *PASP*, vol. 112, pp. 555–565, Apr. 2000.

A. C. S. Readhead, T. S. Nakajima, T. J. Pearson, G. Neugebauer, J. B. Oke, and W. L. W. Sargent, "Diffraction-limited imaging with ground-based optical telescopes," *AJ*, vol. 95, pp. 1278–1296, Apr. 1988.

P. Tuthill, S. Lacour, P. Amico, M. Ireland, B. Norris, P. Stewart, T. Evans, A. Kraus, C. Lidman, E. Pompei, and N. Kornweibel, "Sparse aperture masking (SAM) at NAOS/CONICA on the VLT," in *Ground-based and Airborne Instrumentation for Astronomy III*, ser. Proc. SPIE, vol. 7735, Jul. 2010, p. 77351O.

B. R. M. Norris, P. G. Tuthill, M. J. Ireland, S. Lacour, A. A. Zijlstra, F. Lykou, T. M. Evans, P. Stewart, and T. R. Bedding, "A close halo of large transparent grains around extreme red giant stars," *Nature*, vol. 484, pp. 220–222, Apr. 2012.

M. J. Ireland, P. G. Tuthill, J. Davis, and W. Tango, "Dust scattering in the Miras R Car and RR Sco resolved by optical interferometric polarimetry," *MNRAS*, vol. 361, pp. 337–344, Jul. 2005.

P. Kervella, E. Lagadec, M. Montargès, S. T. Ridgway, A. Chiavassa, X. Haubois, H.-M. Schmid, M. Langlois, A. Gallenne, and G. Perrin, "The close circumstellar environment of Betelgeuse. III. SPHERE/ZIMPOL imaging polarimetry in the visible," *A&A*, vol. 585, p. A28, Jan. 2016.

B. Norris, G. Schworer, P. Tuthill, N. Jovanovic, O. Guyon, P. Stewart, and F. Martinache, "The VAMPIRES instrument: imaging the innermost regions of protoplanetary discs with polarimetric interferometry," *MNRAS*, vol. 447, pp. 2894–2906, Mar. 2015.

J.-L. Beuzit, M. Feldt, K. Dohlen, D. Mouillet, P. Puget, F. Wildi, L. Abe, J. Antichi, A. Baruffolo, P. Baudoz, A. Boccaletti, M. Carbillet, J. Charton, R. Claudi, M. Downing, C. Fabron, P. Feautrier, E. Fedrigo, T. Fusco, J.-L. Gach, R. Gratton, T. Henning, N. Hubin, F. Joos, M. Kasper, M. Langlois, R. Lenzen, C. Moutou, A. Pavlov, C. Petit, J. Pragt, P. Rabou, F. Rigal, R. Roelfsema, G. Rousset, M. Saisse, H.-M. Schmid, E. Stadler, C. Thalmann, M. Turatto, S. Udry, F. Vakili, and R. Waters, "SPHERE: a 'Planet Finder' instrument for the VLT," in *Ground-based and Airborne Instrumentation for Astronomy II*, ser. Proc. SPIE, vol. 7014, Jul. 2008, p. 701418.

B. A. Macintosh, J. R. Graham, D. W. Palmer, R. Doyon, J. Dunn, D. T. Gavel, J. Larkin, B. Oppenheimer, L. Saddlemyer, A. Sivaramakrishnan, J. K. Wallace, B. Bauman, D. A. Erickson, C. Marois, L. A. Poyneer, and R. Soummer, "The Gemini Planet Imager: from science to design to construction," in *Adaptive Optics Systems*, ser. Proc. SPIE, vol. 7015, Jul. 2008, p. 701518.

P. Martinez, E. Aller Carpentier, and M. Kasper, "Laboratory Demonstration of Efficient XAO Coronagraphy in the Context of SPHERE," *PASP*, vol. 122, p. 916, Aug. 2010.