

# OOF Software

## Its Structure and How to Extend It

B. Nikolic

MRAO, Cavendish Laboratory/Kavli Institute for Cosmology  
University of Cambridge

September 2010



**UNIVERSITY OF  
CAMBRIDGE**

# Outline

- 1 Software structure
- 2 Extending OOF software for application at a new telescope
- 3 Interpreting the results

# Outline

- 1 Software structure
  - Mixed C++/Python architecture
  - Modular approach
- 2 Extending OOF software for application at a new telescope
  - Preparing the input files
  - Calculation of the effect of de-focus
- 3 Interpreting the results

# Mixing C++ and Python

## In C++

- 1 All of the algorithms
- 2 Definitions of all data structures

## In Python

- 1 Top level flow control
- 2 Some Input/Output

## Python $\longleftrightarrow$ C++: SWIG

- 1 The binding is automatically generated by SWIG
- 2 Standard way of doing this!

# Motivation for mixed C++ and Python

- Python is interactive:
  - 1 Can easily inspect or change the data structures at any time
  - 2 Easy and flexible control of algorithm parameters
  - 3 Can build more complex algorithms from routines available in C++ code
  - 4 Excellent environment to interactively test C++ code
- Good Input/Output support (including FITS and HDF5)
- Good plotting (matplotlib and PyX)
- This combination is now becoming the standard in astronomy....

# Outline

- 1 Software structure
  - Mixed C++/Python architecture
  - Modular approach
- 2 Extending OOF software for application at a new telescope
  - Preparing the input files
  - Calculation of the effect of de-focus
- 3 Interpreting the results

# Modules

## BNMin1: generalised minimisation and inference

- 1 Non-linear least-squares fitting (primarily Levenberg-Marquardt)
- 2 Markov Chain Monte Carlo (Metropolis algorithm)
- 3 Nested Sampling

## AstroMap: Support for two-dimensional astronomical maps

- 1 Coordinate systems
- 2 Re-gridding data
- 3 FFTs, convolution

## OOF: The application module which brings everything together

- 1 Description of the effect of defocus
- 2 ...

# Auxiliary modules

BNFits: FITS file handling

BNLib: General utility routines

# Outline

- 1 Software structure
- 2 Extending OOF software for application at a new telescope
- 3 Interpreting the results

# Outline

- 1 Software structure
  - Mixed C++/Python architecture
  - Modular approach
- 2 Extending OOF software for application at a new telescope
  - Preparing the input files
  - Calculation of the effect of de-focus
- 3 Interpreting the results

# OOF Input Tasks

- 1 Read data from the telescope specific format
- 2 Compute the offset from the centre of source of each sample of the beam (antenna coordinate system of course!)
- 3 Remove atmospheric signal and any obvious instrumental effects, e.g., by:
  - Fitting a linear trend to end of scans
  - Using a line-free channels when doing spectral line
- 4 Compute noise estimate for each sample
- 5 **Do not regrid the data!**

# OOF Input format

## Standard FITS file format

### Primary header

No data: just keywords specifying basic parameters (telescope name, observing wavelength, etc)

### 1st Extension HDU: Binary table

Data for in-focus map:

Dx	Dy	Fnu	dFnu	Time
...	...	...	...	(Not used)

### 2nd Extension HDU: Binary table

Data for first out-of focus map

Dx	Dy	Fnu	dFnu	Time
...	...	...	...	...

# Available scripts

- 1 Python fragments to write the FITS file with binary tables using the `PyFITS` library
- 2 Fitting to linear trend to end-of-scan (T. Hunter)
- 3 Creating of simulated time series and corresponding FITS files

# Outline

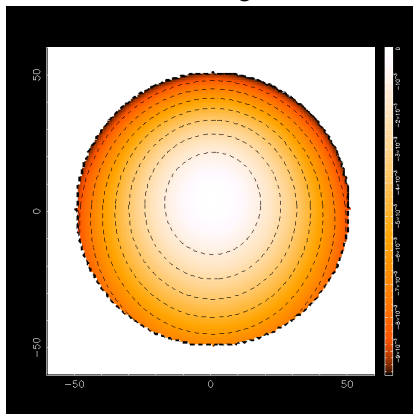
- 1 Software structure
  - Mixed C++/Python architecture
  - Modular approach
- 2 Extending OOF software for application at a new telescope
  - Preparing the input files
  - Calculation of the effect of de-focus
- 3 Interpreting the results

# Calculation of the effect of de-focus

I.e.: How does the aperture plane phase change when the sub-reflector is move?

Offset Gregorian

Simple Cassegrain



# How to implement

- Current implementations are all in C++
- See `oof/src/telgeo/cassegrain.hxx` and `oof/src/telgeo/gbtgeo.hxx`
- Correct routine chosen on basis of telescope name in `oof/src/telgeo/telswitch.hxx` (Add the new telescope here!)
- A prime focus also exists (`oof/src/telgeo/primefoc.hxx`) but so far only used for simulation/testing

# Outline

- 1 Software structure
- 2 Extending OOF software for application at a new telescope
- 3 Interpreting the results**

# Interpreting the results

- 1 **Look** to see if the best-fit model beams look like the observed beams (plots are automatically generated)
- 2 Look at the  $\chi^2$  value and compare to number of input data points
- 3 Look at the correlation matrix (`cvmatrix.csv`) – are any of the parameters very degenerate?
- 4 Look at the retrieved surface
- 5 Repeat this process starting with maximum Zernike order one (i.e., just pointing) and up to 5 or 6 for typical applications