ACHIEVE

0.1

Generated by Doxygen 1.8.11

Contents

Chapter 1

Main Page

Program to read u-v tracks and complex visibilities from an interferometric array and generate an image based on a spherical FT decomposition.

To compile: make

To use: TBD

Add some more detailed descriptions

To see other options, use: ./basis -h

Author

Dimitrios Psaltis

Version

1.0

Date

January 16, 2016

Bug No known bugs

Warning

No known warnings

Todo nothing left

Chapter 2

Todo List

Global ArrayPad (int Ny, int Nx, int Npad, int ∗iRowStart, int ∗iColStart, int ∗NyPad, int ∗NxPad)
nothing left

File fitscopy.c
nothing left

File image2uv.c
Change call to FFTW to use only routines for Real data. Add error capture if-statements for all the read and the write commands

File imarith.c
nothing left

File imcopy.c
nothing left

File imlist.c
nothing left

File imstat.c
nothing left

Global indexArr (int i, int j, int Ny, int Nx)
nothing left

File io.c
nothing left

File listhead.c
nothing left

File liststruc.c
nothing left

page Main Page
nothing left

nothing left

Global makeUVmodel (int iModel, double param[], int Nu, int Nv, double Uco[], double Vco[], double Vis↩ Amp[NuMax][NvMax], double VisPhase[NuMax][NvMax])
add more models; for now it only has the crescent model

File modelsImage.c
Nothing left

File modhead.c

nothing left

Global readFITSImage (char fname[], int Ny, int Nx, int Npad, double ∗Image)

nothing left

Global readFITSImagedim (char fname[], int ∗Ny, int ∗Nx, double ∗yScale, double ∗xScale)

nothing left

File synthimage.c

Nothing to do

File tabcalc.c

nothing left

File tablist.c

nothing left

File tabmerge.c

nothing left

Global uvToPolar (int Nu, int Nv, int Nk, int Nm, double Uco[], double Vco[], double VisAmp[NuMax][Nv←↩Max], double VisPhase[NuMax][NvMax], double kco[], double RePk[NkMax][NmMax], double ImPk[Nk←↩Max][NmMax])

It performs the phi integration given NPHI=128 points. In principle, this should depend on k. Given that the u-v grid is Cartesian, NPHI should be small at low k and large at large k.

Global writeFITSVis (char fname[], int Ny, int Nx, double ∗Vp, double ∗Va, double vScale, double uScale, char hist[])

nothing left

Chapter 3

Bug List

Global ArrayPad (int Ny, int Nx, int Npad, int ∗iRowStart, int ∗iColStart, int ∗NyPad, int ∗NxPad)
  No known bugs

File fitscopy.c
  No known bugs

File image2uv.c
  No known bugs

File imarith.c
  No known bugs

File imcopy.c
  No known bugs

File imlist.c
  No known bugs

File imstat.c
  No known bugs

Global indexArr (int i, int j, int Ny, int Nx)
  No known bugs

File io.c
  No known bugs

File listhead.c
  No known bugs

File liststruc.c
  No known bugs

page Main Page
  No known bugs

  No known bugs

File modelsImage.c
  No known bugs

File modhead.c
  No known bugs

Global readFITSImage (char fname[], int Ny, int Nx, int Npad, double ∗Image)
  No known bugs

Global readFITSImagedim (char fname[], int ∗Ny, int ∗Nx, double ∗yScale, double ∗xScale)

No known bugs

File synthimage.c

No known bugs

File tabcalc.c

No known bugs

File tablist.c

No known bugs

File tabmerge.c

No known bugs

Global writeFITSVis (char fname[], int Ny, int Nx, double ∗Vp, double ∗Va, double vScale, double uScale, char hist[])

No known bugs

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 my_plot Namespace Reference

my_plot contains the main plotting functions which are used by the rest of the modules.

6.1.1 Detailed Description

my_plot contains the main plotting functions which are used by the rest of the modules.

Specifically, it contains functions plot VLBI observables fromm GRMHD simulations.

Chapter 7

File Documentation

7.1 Fits/fitscopy.c File Reference

Copies on image file to another, while applying complex filtering methods.

```
#include <stdio.h>
#include "fitsio.h"
```

Functions

- int **main** (int argc, char *argv[ ])

7.1.1 Detailed Description

Copies on image file to another, while applying complex filtering methods.

To use: fitscopy infile[ext][filters] outfile

This seemingly simple program can apply complex filtering methods which transform the input file as it is copied. For example, it can select a subimage out of a larger image, select rows in a table based on a logical expression, create new table columns or modify the values in existing columns, and filter tables based on time (using GTI extensions) or on spatial position (using spatial region files). See the Extended File Name syntax page for more examples of the filtering syntax, and refer to the CFITSIO User's Guide for a complete description. Note that in these examples it is often necessary to enclose the input filename in single quote characters on the Unix command line if the name contains special characters such as '[' or '*'.

EXAMPLES

- fitscopy in.fit out.fit - simple file copy
- fitscopy in.fit \!out.fit - overwrite out.fit file

(In Unix, the '!' must be preceded by a '\' on the command line)

- fitscopy 'in.fit[11:50,21:50]' out.fit - copy 40x30 pixel subimage
- fitscopy 'in.fit[-*,*]' out.fit - mirror reverse the image
- fitscopy 'stdin[11:50,21:60]' stdout - piped subimage
- fitscopy iniraf.imh out.fit - convert IRAF image to FITS
- fitscopy 'in.dat[i512,512]' out.fit - binary file to FITS
- fitscopy 'in.fit[evt][pi>35]' out.fit - copy rows which have pi>35
- fitscopy 'in.fit[2][bin X,Y]' out.fit - bin X,Y cols to make image
- fitscopy 'in.fit[evt][col Xc=3.*X]' out.fit - create new Xc column
- fitscopy 'in.fit[evt][gtifilter()]' out.fit - apply GTI time filter
- fitscopy 'in.fit[2][regfilter("pow.reg")]' out.fit - region filter

The program itself is almost trivial, with just 6 lines of code required to open the input file, create the empty output file, copy each HDU from the input file to the output file, then close the 2 files. (See a listing of fitscopy.c).

Version 1.0: September 29, 2017; as is

Author

Dimitrios Psaltis (updated from `https://heasarc.gsfc.nasa.gov/docs/software/fitsio/cexamples.↩ html`)

Version

1.0

Date

September 29, 2017

Bug No known bugs

Warning

no warnings

Todo nothing left

7.2 Fits/imarith.c File Reference

Performs arithmetic operations involving one or two images.

```
#include <string.h>
#include <stdio.h>
#include "fitsio.h"
```

Functions

- int **main** (int argc, char ∗argv[ ])

7.2.1 Detailed Description

Performs arithmetic operations involving one or two images.

Use: imarith image1[ext] image2[ext] operation outimage (2 images)

or

imarith image[ext] value operation outimage (1 image)

Add, subtract, multiply, or divide the pixels in one image by another image or a constant and write the result to a new output image. Supported arithmetic operations are 'add', 'sub', 'mul', or 'div' (only the first character need be supplied)

Examples:

- imarith in1.fit in2.fit add out.fit - add the 2 images
- imarith in.fit 1.2 mul out.fit - multiply image by 1.1
- imarith 'in1.fit[1:20,1:20]' 'in2.fit[1:20,1:20]' add !out.fit

  add 2 image sections; also overwrite out.fit if it exists.
- imarith data.fit[1] data.fit[2] mul out.fit - multiply the images

  in the 1st and 2nd extensions of the file data.fit

This program first opens the input images. If 2 images, it checks that they have the same dimensions. It then creates the empty output file and copies the header of the first image into it (thus duplicating the size and data type of that image). Memory is allocated to hold one row of the images. The program computes the output image values by reading one row at a time from the input image(s), performing the desired arithmetic operation, and then writing the resulting row to the output file. The program reads and writes the row of data in double precision format, regardless of the intrinsic datatype of the images; CFITSIO transparently converts the data format if necessary as the rows are read and written. This program also supports 3D data cubes by looping through each plane of the cube.

2010-08-19 modified to allow numeric second argument (contributed by Michal Szymanski, Warsaw University Observatory)

Version 1.0: September 29, 2017

Author

Dimitrios Psaltis (updated from `https://heasarc.gsfc.nasa.gov/docs/software/fitsio/cexamples.↩ html`)

Version

1.0

Date

September 29, 2017

Bug No known bugs

Warning

no warnings

Todo nothing left

7.3 Fits/imcopy.c File Reference

Copy the input FITS image to the new output file with optional data compression.

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "fitsio.h"
```

Functions

- int **main** (int argc, char ∗argv[ ])

7.3.1 Detailed Description

Copy the input FITS image to the new output file with optional data compression.

Use: imcopy inimage[ext] outimage[compress]

Copy the input FITS image to the new output file with optional data compression. The output image maybe externally compressed with the gzip or Unix compress algorithm, or it may be internally compressed with the new tile compression format. The output image may be written in the tile-compressed FITS image format by adding the compression qualifiers in square brackets following the output file name. Examples:

- imcopy infile.fit 'outfile.fit[compress]'

This will use the default compression algorithm (Rice) and the default tile size (row by row)

- imcopy infile.fit 'outfile.fit[compress GZIP]'

This will use the GZIP compression algorithm and the default tile size (row by row). The allowed compression algorithms are Rice, GZIP, and PLIO. Only the first letter of the algorithm name needs to be specified.

- imcopy infile.fit 'outfile.fit[compress G 100,100]'

This will use the GZIP compression algorithm and 100 X 100 pixel tiles.

- imcopy infile.fit 'outfile.fit[compress R 100,100; 4]'

This will use the Rice compression algorithm, 100 X 100 pixel tiles, and noise_bits = 4 (assuming the input image has a floating point data type). Decreasing the value of noisebits will improve the overall compression efficiency at the expense of losing more information.

- imcopy infile.fit outfile.fit

If the input file is in tile-compressed format, then it will be uncompressed to the output file. Otherwise, it simply copies the input image to the output image.

- imcopy 'infile.fit[1001:1500,2001:2500]' outfile.

This extracts a 500 X 500 pixel section of the much larger input image (which may be in tile-compressed format). The output is a normal uncompressed FITS image.

- imcopy 'infile.fit[1001:1500,2001:2500]' outfile.fit.gz

Same as above, except the output file is externally compressed using the gzip algorithm.

Version 1.0: September 29, 2017

Author

Dimitrios Psaltis (updated from `https://heasarc.gsfc.nasa.gov/docs/software/fitsio/cexamples.↩ html`)

Version

1.0

Date

September 29, 2017

Bug No known bugs

Warning

no warnings

Todo nothing left

7.4 Fits/imlist.c File Reference

Prints pixel values in all or part of an image.

```
#include <string.h>
#include <stdio.h>
#include "fitsio.h"
```

Functions

- int **main** (int argc, char ∗argv[ ])

7.4.1 Detailed Description

Prints pixel values in all or part of an image.

List the pixel values in all or part of an image. In order to reduce the amount of output, it is usually more useful to print only a small section of the image by specifying the minimum and maximum pixel in each dimension to be printed, enclosed in square brackets following the root file name and optional extension specifier.

Examples:

- imlist infile.fit - print the whole image
- imlist 'in.fit[∗:10,∗:10]' - print every 10th pixel
- imlist 'in.fit[20:29,10:19]' - print 10x10 section
- imlist 'in.fit[20:29:2,10:19:2]' - print every other pixel
- imlist 'in.fit[3][20:29,10:19]' - print 3rd IMAGE extension
- imlist 'intab.fit[1][bin (X,Y)=32] - print image generated by binning the X and Y table columns with a binning size = 32

This program opens the file (which may have been filtered) and checks that the HDU is a 1D or 2D image. It then allocates memory to hold 1 row of pixels, and loops through the image from top to bottom reading and printing out each row. The data format of the image (integer or floating point) determines the output display format that is used for the pixel values. Note that the program reads the image into an array of doubles, regardless of the intrinsic datatype of the image.

Version 1.0: September 29, 2017 (DP)

Changed format for printing from "%f" to "%e"

Author

Dimitrios Psaltis (updated from `https://heasarc.gsfc.nasa.gov/docs/software/fitsio/cexamples.↩ html`)

Version

1.0

Date

September 29, 2017

Bug No known bugs

Warning

no warnings

Todo nothing left

7.5 Fits/imstat.c File Reference

Computes simple statistics for the pixels in an input image.

```
#include <string.h>
#include <stdio.h>
#include "fitsio.h"
```

Functions

- int **main** (int argc, char ∗argv[ ])

7.5.1 Detailed Description

Computes simple statistics for the pixels in an input image.

Use: imstat infile[ext]

Computes simple statistics for the pixels in an input image. It computes the sum of all the pixels, the mean pixel value, and the minimum and maximum pixel values.

Examples:

- imstat infile.fit - stats of the whole image
- imstat 'in.fit[20:29,10:19]' - stats of 10x10 pixel section
- imstat 'in.fit[3][20:29,10:19]' - stats of 3rd IMAGE extension
- imstat 'intab.fit[1][bin (X,Y)=32] - stats of an image generated by binning the X and Y table columns with a binning size = 32

This program opens the file (which may have been filtered) and checks that the HDU is a 2D image. (It could easily be extended to support 3D data cubes as well, as is done in the imarith program). It then allocates memory to hold 1 row of pixels, and loops through the image, accumulating the statistics row by row. Note that the program reads the image into an array of doubles, regardless of the intrinsic datatype of the image.

Version 1.0: September 29, 2017 (DP)

Added brightness center and location of min and max brightness

Author

Dimitrios Psaltis (updated from `https://heasarc.gsfc.nasa.gov/docs/software/fitsio/cexamples.↩ html`)

Version

1.0

Date

September 29, 2017

Bug No known bugs

Warning

no warnings

Todo nothing left

7.6 Fits/listhead.c File Reference

Lists the header keywords in the specified HDU (Header Data Unit) of a file.

```
#include <string.h>
#include <stdio.h>
#include "fitsio.h"
```

Functions

- int **main** (int argc, char ∗argv[ ])

7.6.1 Detailed Description

Lists the header keywords in the specified HDU (Header Data Unit) of a file.

Use: listhead infile[ext]

This program will list the header keywords in the specified HDU (Header Data Unit) of a file. If a HDU name or number is not appended to the input root file name, then the program will list the keywords in every HDU in the file.

EXAMPLES;

- listhead file.fit - list all the headers in the file
- listhead 'file.fit[0]' - list the primary array header
- listhead 'file.fit[2]' - list the header of 2nd extension
- listhead file.fit+2 - same as above
- listhead 'file.fit[GTI]' - list header of the 'GTI' extension

This program is very short. It simply opens the input file and then reads and prints out every keyword in the current HDU in sequence. If a specific HDU was not specified as part of the input file name, it then trys to move to the next HDU in the file and list its keywords. This continues until it reaches the end of the file.

Version 1.0: September 29, 2017 (DP)

Author

Dimitrios Psaltis (updated from `https://heasarc.gsfc.nasa.gov/docs/software/fitsio/cexamples.↩ html`)

Version

1.0

Date

September 29, 2017

Bug No known bugs

Warning

no warnings

Todo nothing left

7.7 Fits/liststruc.c File Reference

Lists the organizational structure of the specified HDU of a file.

```
#include <string.h>
#include <stdio.h>
#include "fitsio.h"
```

Functions

- int **main** (int argc, char ∗argv[ ])

7.7.1 Detailed Description

Lists the organizational structure of the specified HDU of a file.

Use: liststruc infile[ext]

This program will list the organizational structure of the specified HDU of a file. If a HDU name or number is not appended to the input root file name, then the program will list the structure of every HDU in the file. If the HDU is the primary array or IMAGE extension then it lists the dimensions and datatype of the image. Otherwise, if the HDU is a table then it lists the name and data format of all the columns in the table.

EXAMPLES:

- liststruc file.fit - list structure of every HDU in the file
- liststruc 'file.fit[0]' - list structure of the primary array

This program opens the file and determines the type (i.e., image or table) of the current HDU. The relevant structural parameters for that HDU are then printed out. If a specific HDU is not given as part of the input file name, then the program moves to successive HDUs in the file, if any, and repeats the process until it reaches the end of the file.

Version 1.0: September 29, 2017 (DP)

Changed a little the format of the output

Author

Dimitrios Psaltis (updated from `https://heasarc.gsfc.nasa.gov/docs/software/fitsio/cexamples.↩ html`)

Version

1.0

Date

September 29, 2017

Bug No known bugs

Warning

no warnings

Todo nothing left

7.8 Fits/modhead.c File Reference

Writes or Modifies the value of a header keyword.

```
#include <string.h>
#include <stdio.h>
#include "fitsio.h"
```

Functions

- int **main** (int argc, char ∗argv[ ])

7.8.1 Detailed Description

Writes or Modifies the value of a header keyword.

Use: modhead infile[ext] keyword newvalue

This program can write or modify the value of a header keyword. If the 4th 'newvalue' parameter is not specified then the program just prints out the current value of the keyword. Otherwise, it replaces the value of the keyword if it exists, or writes a new keyword if it doesn't exist.

Examples:

- modhead file.fits dec - list the value of the DEC keyword
- modhead file.fits dec 30.0 - write or modify the DEC keyword

This program opens the input file and attempts to read the specified keyword. If the keyword exists, the current value is printed out. If the 4th 'newvalue' argument is specified then the program first checks to make sure that we are not attempting to modify a structural keyword (like NAXIS) which could corrupt the file format. It then constructs a new keyword record string with the new value and overwrites or appends that record to the header.

Version 1.0: September 29, 2017 (DP)

Author

Dimitrios Psaltis (updated from `https://heasarc.gsfc.nasa.gov/docs/software/fitsio/cexamples.↩ html`)

Version

1.0

Date

September 29, 2017

Bug No known bugs

Warning

no warnings

Todo nothing left

7.9 Fits/tabcalc.c File Reference

Computes new values for the specified table column using the input arithmetic expression.

```
#include <string.h>
#include <stdio.h>
#include "fitsio.h"
```

Functions

- int **main** (int argc, char ∗argv[ ])

7.9.1 Detailed Description

Computes new values for the specified table column using the input arithmetic expression.

Use: tabcalc intable expression colname outtable

Computes new values for the specified table column using the input arithmetic expression, which may be a function of the values in other table columns or keyword values. The input file is first copied to the output file, then the output file is updated with the new column values. If the column doesn't already exist, then a new column will be appended to the table. Instead of supplying the arithmetic expression directly on the command line, you can also put the expression in an ASCII text file, and supply the name of the text file prepended with the '@' character.

The following arithmetic operators and functions may be used in the expression:

```
"addition"          +          "subtraction"        -
"multiplication"    *          "division"           /
"negation"          -          "exponentiation"     **   ^
"absolute value"    abs(x)     "cosine"             cos(x)
"sine"              sin(x)     "tangent"            tan(x)
"arc cosine"        arccos(x)  "arc sine"           arcsin(x)
"arc tangent"       arctan(x)  "arc tangent"        arctan2(x,y)
"exponential"       exp(x)     "square root"        sqrt(x)
"natural log"       log(x)     "common log"         log10(x)
"modulus"           i % j      "random # [0.0,1.0)" random()
"minimum"           min(x,y)   "maximum"            max(x,y)
"if-then-else"      b?x:y
```

Note that the fitscopy program can perform the same operations as tabcalc, using a slightly different command line syntax.

Examples:

- tabcalc in.fit+1 'X+0.5' Xc out.fit - calculate new Xc column

  fitscopy 'in.fit+1[col Xc=X+0.5]' out.fit - same as above
- tabcalc in.fit+1 .txt Xc out.fit - read expression from file

Other example expressions:

'counts/#exposure' - divide counts column by EXPOSURE keyword

'counts $>$ 0 ? counts/#exposure : -99' - if counts column is greater than 0 then divide counts by the exposure keyword, otherwise set the value equal to -99.

'(counts{-1} $*$ 0.25 + counts $*$ 0.5 + counts{+1} $*$ 0.25) / 3.' compute weighted running mean of counts by averaging the values in the previous row, the current row, and the next row. The row offset is enclosed in curly brackets.

Version 1.0: September 29, 2017 (DP)

Author

Dimitrios Psaltis (updated from https://heasarc.gsfc.nasa.gov/docs/software/fitsio/cexamples.↩
html)

Version

1.0

Date

September 29, 2017

Bug No known bugs

Warning

no warnings

Todo nothing left

7.10 Fits/tablist.c File Reference

Lists the values in all the rows and columns of the input table.

```
#include <string.h>
#include <stdio.h>
#include "fitsio.h"
```

Functions

- int **main** (int argc, char ∗argv[ ])

7.10.1 Detailed Description

Lists the values in all the rows and columns of the input table.

Use: tablist infile[ext][col filter][rowfilter]

Lists the values in all the rows and columns of the input table. In order to reduce the amount of output, it is sometimes useful to select only certain rows or certain columns in the table to be printed by filtering the input file as shown in the examples.

Examples:

- tablist 'intab.fit[EVENTS]' - print 'EVENTS' table
- tablist 'intab.fit[1]' - print 1st extension

- tablist 'intab.fit[1][col X;Y]' - print X and Y columns
- tablist 'intab.fit[1][PHA > 24]' - print rows with PHA > 24
- tablist 'intab.fit[1][col X;Y][PHA > 24]' - as above
- tablist 'intab.fit[1][col X;Y;Rad=sqrt(X**2+Y**2)] - print X, Y, and a new 'Rad' column that is computed on the fly.

This program opens the file (which may have been filtered to select only certain rows or columns in the table) and checks that the HDU is a table. It then calculates the number of columns that can be printed on an 80-column output line. The program reads and prints out the values in these columns row by row. Once all the rows have been printed out, it repeats the process for the next set of columns, if any, until the entire table has been printed. All the column values are read as formatted character strings, regardless of the intrinsic datatype of the column. CFITSIO uses the TDISPn keyword if it exists to format the string (e.g., TDISP1 = 'F8.2', or TDISP2 = 'E15.6'), otherwise it uses a default display format for each column datatype. It should be cautioned that reading a table this way, one element at a time column by column and row by row, is not very efficient. For large tables, it is more efficient to read each column over least 100 rows at a time into temporary arrays. After processing all these rows, then read the next 100 rows from the table, and so on, until all the rows have been processed.

Version 1.0: September 29, 2017 (DP)

Author

Dimitrios Psaltis (updated from `https://heasarc.gsfc.nasa.gov/docs/software/fitsio/cexamples.↩ html`)

Version

1.0

Date

September 29, 2017

Bug No known bugs

Warning

no warnings

Todo nothing left

7.11 Fits/tabmerge.c File Reference

Merges 2 tables by appending all the rows from the 1st table onto the 2nd table.

```
#include <string.h>
#include <stdio.h>
#include "fitsio.h"
```

Functions

- int **main** (int argc, char *argv[ ])

7.11.1 Detailed Description

Merges 2 tables by appending all the rows from the 1st table onto the 2nd table.

Use: tabmerge intable[ext][filters] outtable[ext]

Merge 2 tables by appending all the rows from the 1st table onto the 2nd table. The 2 tables must have identical structure, with the same number of columns with the same datatypes.

Examples:

- tabmerge in.fit+1 out.fit+2 append the rows from the 1st extension of the input file into the table in the 2nd extension of the output file.
- tabmerge 'in.fit+1[[PI $>$ 45]' out.fit+2 Same as the 1st example, except only rows that have a PI column value $>$ 45 will be merged into the output table.

In this program, a series of 'if' statements are used to verify that the input files exist and that they point to valid tables with identical sets of columns. If all these checks are satisfied correctly, then additional empty rows are appended to the end of the output table, and the rows are copied one by one from the input table to the output table. Since the tables have identical structures, it is possible here to use the low-level CFITSIO routines that read and write each row of the table as a raw string of bytes. This is much faster than having to read and write the numerical values in each column individually.

Note that in this example the output file is modified in place, rather than creating a whole new output file. It is often easier to modify the existing file in this way rather than create a new one, but there is a slight risk that the output file could be corrupted if the program crashes before the modifications have been completed (e.g., due to a power failure, or the user exits the program with Cntl-C, or there is insufficient disk space to write the modified file).

Version 1.0: September 29, 2017 (DP)

Author

Dimitrios Psaltis (updated from `https://heasarc.gsfc.nasa.gov/docs/software/fitsio/cexamples.↩ html`)

Version

1.0

Date

September 29, 2017

Bug No known bugs

Warning

no warnings

Todo nothing left

7.12 image.c File Reference

```
#include "definitions.h"
#include "global.h"
```

Functions

- double polarAngle (double x, double y)

  Given two Cartesian coordinates (x,y), it returns the polar angle phi of the corresponding position vector, in the range 0->2∗pi.
- double bessj0 (double x)

  Returns the Bessel function J_0(x) for any real x.
- double bessj (int n, double x)

  Returns the Bessel function J_n(x) for any real x.
- int makeImageGrid (int Na, int Nb, double amax, double bmax, double aco[ ], double bco[ ])

  Generates a grid of coordinates along the image plane.
- int polarToImage (int Nk, int Nm, int Na, int Nb, int kmin, int kmax, int mmin, int mmax, double kco[ ], double RePk[NkMax][NmMax], double ImPk[NkMax][NmMax], double aco[ ], double bco[ ], double Image[NAM←AX][NBMAX])

  Converts a polar k-m map to an image.

7.12.1 Function Documentation

7.12.1.1 double bessj (int *n*, double *x*)

Returns the Bessel function J_n(x) for any real x.

(DP) Changed all floats to doubles; Also calls the relevant Bessel functions for n=0 and n=1

Author

Numerical Recipes in C, 2nd edition, pg. 232

Version

1.0

Date

February 8, 2016

Parameters

n	an integer argument with the order of the Bessel function
x	a double argument for the Bessel function

Returns

Returns the value of the bessel function

7.12.1.2 double bessj0 (double *x*)

Returns the Bessel function J_0(x) for any real x.

(DP) Changed all floats to doubles

Author

Numerical Recipes in C, 2nd edition, pg. 232

Version

1.0

Date

February 8, 2016

Parameters

x	a double argument for the Bessel function

Returns

Returns the value of the bessel function

7.12.1.3 int makeImageGrid (int *Na,* int *Nb,* double *amax,* double *bmax,* double *aco[ ],* double *bco[ ]*)

Generates a grid of coordinates along the image plane.

It generates an equidistant grid of Na alpha-points and of Nb beta-points and stores the two grids in the double arrays aco[] and bco[]. The grid are from -amax -> amax and from -bmax ->bmax. Both Na and Nb need to be odd. The routine checks whether the number of grid points is less than the maximum numbers NAMAX and NBMAX stored in the global definitions.

Author

Dimitrios Psaltis

Version

1.0

Date

February 19, 2016

Parameters

Na	an int with the number of grid points in the alpha-direction
Nb	an int with the number of grid points in the beta-direction
amax	a double with the maximum value of the alpha coordinate
bmax	a double with the maximum value of the beta coordinate
aco[ ]	a double array where the alpha coordinates will be stored
bco[ ]	a double array where the beta coordinates will be stored

Returns

0 if it was successful, 1 if there was an error, e.g., too many grid points or not an odd number

7.12.1.4 double polarAngle (double *x*, double *y*)

Given two Cartesian coordinates (x,y), it returns the polar angle phi of the corresponding position vector, in the range 0->2∗pi.

Useful in converting from Cartesian to Polar Coordinates

Author

Dimitrios Psaltis

Version

1.0

Date

February 19, 2016

Parameters

x	a double argument for the horizontal Cartesian coordinate
y	a double argument for the vertical Cartesian coordinate

Returns

Returns the value of the phi angle in radians

7.12.1.5 int polarToImage (int *Nk*, int *Nm*, int *Na*, int *Nb*, int *kmin*, int *kmax*, int *mmin*, int *mmax*, double *kco[ ]*, double *RePk[NkMax][NmMax]*, double *ImPk[NkMax][NmMax]*, double *aco[ ]*, double *bco[ ]*, double *Image[NAMAX][NBMAX]*)

Converts a polar k-m map to an image.

Given a grid of (Nk,Nm) points of the polar transform (with the real part stored in RePk[][] and the imaginary part in ImPk[][]), it returns the corresponding image on a (Na,Nb) grid of points, with the grid of alpha and beta points stored in aco[] and bco[] and the image stored in Image[][].

It will only use the k-modes between kmin and kmax as well as the m-modes between mmin and mmax, as given in the inputs

It uses the trapezoid rule to perform the integral over k and a simple summation for m. NB: it assumes that the k-coordinates are equidistant.

Author

Dimitrios Psaltis

Version

1.0

Date

February 19, 2016

Parameters

Nk	an int with the number of grid points in the k-direction
Nm	an int with the number of grid points in the m-direction
Na	an int with the number of grid points in the alpha-direction
Nb	an int with the number of grid points in the beta-direction
kmin	an int with the minimum k-mode to be used
kmax	an int with the maximum k-mode to be used
mmin	an int with the minimum m-mode to be used
mmax	an int with the maximum m-mode to be used
kco[ ]	a double array with the k-coordinates
RePk[ ][ ]	a double array in which the real parts of the polar transform will be stored
ImPk[ ][ ]	a double array in which the imaginary parts of the polar transform will be stored
aco[ ]	a double array with the alpha-coordinates
bco[ ]	a double array with the beta-coordinates
Image[ ][ ]	a double array with the image brightness

Returns

0 if it was successful, 1 if there was an error

7.13 image2uv.c File Reference

Converts an image to u-v maps (complex amplitude and phase)

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <fftw3.h>
#include "fitsio.h"
```

Macros

- #define DEFAULTOUTFILENAME "uvout.fits"

  default filename for output, if -o option is not given
- #define VMODEDEFAULT 1

  default verbose mode "medium"
- #define MAXCHAR 80

  maximum number of characters for strings
- #define RED "\x1B[31m"

  color RED for error output
- #define RESETCOLOR "\x1B[0m"

  color to reset to normal
- #define MINAMP 1.e-12

  minimum fraction of zero baseline amplitude, below which the phase is set to zero

Functions

- void printErrorImage2uv (char errmsg[ ])

  Prints an error message.
- void printhelp (void)

  Prints a help message when no other arguments are given.
- void verboseinput (char ∗outFileName, int ∗vmode, int ∗cmode, int ∗Npad)

  Asks for user input for all the options of this command.
- int parse (int argc, char ∗argv[ ], char ∗inFileName, char ∗outFileName, int ∗vmode, int ∗cmode, int ∗Npad)

  Parses the command line for options.
- int main (int argc, char ∗argv[ ])

  Main program.

7.13.1 Detailed Description

Converts an image to u-v maps (complex amplitude and phase)

This program reads an image stored in an input FITS file, calculates its complex Fourier transform, and stores the resulting visibility amplitudes and phases in an output FITS file.

The dimension of the Fourier transform along each axis is the maximum of the dimension of the image along the same axis and the optional number of padding points.

The output FITS file has two HDU images; the first one has the visibility amplitudes and the second one has the visibility phases (in rad). The center of the Fourier transform is at grid point (starting from 1) Nx/2 and Ny/2, where Nx (or Ny) is the maximum of the dimension of the image along the horizontal (vertical) axis and the optional number of padding points.

In order to avoid numerical issues, when the amplitude is smaller than a predefined fraction (MINAMP) of the zero baseline amplitude, the phase is set to zero.

Use: image2uv [-sv] [-p Npoints] [-c] [-o filename2] filename1

The required options are:

- "filename1": sets the input image filename (FITS)

The optional options are:

- "-o filename2": sets the output visibility filename (UVFITS).
- "-s": silent mode. It does not print anything and uses defaults
- "-v": verbose mode. It prints a lot more information
- "-p Npoints": pads the image to a square grid with Npoints on each side, if the current image size is smaller than Npoints, before taking the Fourier Transform
- "-c": calculates the complex phases by first centering the image to its center of brightness. If this options is not given, it calculates the complex phase with respect to the geometric center of the image.

If no options are given, it prints a help message

Examples:

- image2uv -o uvresults.out inimage.fits

Reads the file form inimage.fits and outputs the visibility amplitudes and phases into uvresults.out

Author

Dimitrios Psaltis

Version

1.0

Date

September 30, 2017

Bug No known bugs

Warning

No known warnings

Todo Change call to FFTW to use only routines for Real data. Add error capture if-statements for all the read and the write commands

7.13.2 Function Documentation

7.13.2.1 int main (int *argc*, char ∗ *argv[ ]*)

Main program.

Author

Dimitrios Psaltis

Version

1.0

Date

September 30, 2017

Precondition

Nothing

7.13.2.2 int parse (int *argc*, char ∗ *argv[ ]*, char ∗ *inFileName*, char ∗ *outFileName*, int ∗ *vmode*, int ∗ *cmode*, int ∗ *Npad*)

Parses the command line for options.

Parses the command line for options. If no options are given, it prints a help message

The required options are:

- <filename>: sets the input image filename (FITS)

The optional options are:

- "-o <filename>": sets the output visibility filename (UVFITS).
- "-s": silent mode. It does not print anything
- "-v": verbose mode. It prints a lot more information
- "-p Npoints": pading. It pads the image to a square grid with Npoints on each side
- "-c": calculates the complex phase by first centering the image to its center of brightness.

Author

Dimitrios Psaltis

Version

1.0

Date

September 30, 2017

Precondition

It is called from main()

Parameters

argc	an int (as is piped from the unix prompt)
argv[ ]	an array of strings (as is piped from the unix prompt)
∗*inFileName*	a string which returns the input filename
∗*outFileName*	a string which returns the output filename
∗*vmode*	an int with a flag for the chosen verbose mode (0:silent, 1: normal, 2: verbose)
∗*cmode*	an int with a flat for whether the FFT will be centered on the center of brightness
∗*Npad*	an int with the number of points per dimension to which the image will be padded. It it is smaller than the number of points in the image, then the total number of points in the image will be used.

Returns

Returns zero if successful, 1 if not

7.13.2.3 void printErrorImage2uv (char *errmsg[ ]*)

Prints an error message.

Author

Dimitrios Psaltis

Version

1.0

Date

September 30, 2017

Parameters

errmsg[ ]	a string with the error message to be printed

Returns

nothing

7.13.2.4 void printhelp (void)

Prints a help message when no other arguments are given.

Author

Dimitrios Psaltis

Version

1.0

Date

September 30, 2017

Precondition

It is called from parse() and from main()

Parameters

no	parameters

Returns

nothing

7.13.2.5 void verboseinput (char ∗ *outFileName,* int ∗ *vmode,* int ∗ *cmode,* int ∗ *Npad*)

Asks for user input for all the options of this command.

It asks the user to input directly all the possible options that one can give in this command. It is called from the main function if the '-v' option is given as an argument.

When each option is presented, the default value or the value that was passed as an argument is displayed. If the user hits return, then that value remains. If the user provides a valid answer, it replaces the value for that option.

Author

Dimitrios Psaltis

Version

1.0

Date

November 12, 2017

Precondition

It is called from main()

Parameters

∗*outFileName*	a string which provides and returns the output filename
∗*vmode*	an int with a flag for the chosen verbose mode (0:silent, 1: normal, 2: verbose)
∗*cmode*	an int with a flat for whether the FFT will be centered on the center of brightness
∗*Npad*	an int with the number of points per dimension to which the image will be padded. It it is smaller than the number of points in the image, then the total number of points in the image will be used.

Returns

Returns zero if successful, 1 if not

7.14 io.c File Reference

Subroutines to perform I/O with FITS, OIFITS, etc. files.

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include "fitsio.h"
```

Macros

- #define RED "\x1B[31m"

  color RED for error output
- #define RESETCOLOR "\x1B[0m"

  color to reset to normal

Functions

- void printErrorIO (char errmsg[ ])

  Prints an error message.
- int readFITSImagedim (char fname[ ], int ∗Ny, int ∗Nx, double ∗yScale, double ∗xScale)

  Reads the image dimensions of a FITS file, both in terms of pixels and in terms of physical units.
- int readFITSImage (char fname[ ], int Ny, int Nx, int Npad, double ∗Image)

  Reads the image stored in a FITS file and, optionally, pads it with zeros.
- int writeFITSVis (char fname[ ], int Ny, int Nx, double ∗Vp, double ∗Va, double vScale, double uScale, char hist[ ])

  Writes visibility amplitudes and phases into a FITS file.
- int indexArr (int i, int j, int Ny, int Nx)

  Function to convert 2D array indices to a pointer location.
- int ArrayPad (int Ny, int Nx, int Npad, int ∗iRowStart, int ∗iColStart, int ∗NyPad, int ∗NxPad)

  Function to calculate sizes and starting locations of padded images.

7.14.1 Detailed Description

Subroutines to perform I/O with FITS, OIFITS, etc. files.

Author

Dimitrios Psaltis

Version

1.0

Date

October 2, 2017

Bug No known bugs

Warning

No known warnings

Todo nothing left

7.14.2 Function Documentation

7.14.2.1 int ArrayPad (int *Ny,* int *Nx,* int *Npad,* int ∗ *iRowStart,* int ∗ *iColStart,* int ∗ *NyPad,* int ∗ *NxPad*)

Function to calculate sizes and starting locations of padded images.

Given an initial size Ny∗Nx of an immage array and a padding integer Npad, it returns (i) the starting row (iRowStart) and column (iColStart) of the actual image in the padded array (with the first pixel counting from [1,1]) (ii) the size of the padded array along each direction (NxPad and NyPad), which will be the larger of the initial size and Npad, along each direction

It returns zero if it all worked out, one if it didn't.

Parameters

Ny	an int with the number of rows in the image
Nx	an int with the number of columns in the image
Npad	an int with the padding parameter
∗*iRowStart*	an int pointer that returns the starting row of the image in the padded array
∗*iColStart*	an int pointer that returns the starting column of the image in the padded array
NyPad	an int pointer that returns the number of rows of the padded image
NxPad	an int pointer that returns the number of columns of the padded image

Author

Dimitrios Psaltis

Version

1.0

Date

October 3, 2017

Bug No known bugs

Warning

it assumes that all integers are positive, so it never returns anything but zero

Todo nothing left

7.14.2.2 int indexArr (int *i*, int *j*, int *Ny*, int *Nx*)

Function to convert 2D array indices to a pointer location.

Given an [Ny][Nx] array, it takes the two coordinates i and j of an element and converts it to an incremental pointer location. The i- and j- coordinates start from (1,1) for the first point

Parameters

i	an int with the row coordinate of an element
j	an int with the column coordinate of an element
Ny	an int with the number of rows in the image (not used)
Nx	an int with the number of columns in the image

Author

Dimitrios Psaltis

Version

1.0

Date

October 3, 2017

Bug No known bugs

Warning

No known warnings

Todo nothing left

7.14.2.3 void printErrorIO (char *errmsg[ ]*)

Prints an error message.

Author

Dimitrios Psaltis

Version

1.0

Date

September 30, 2017

Parameters

errmsg[ ]	a string with the error message to be printed

Returns

nothing

7.14.2.4 int readFITSImage (char *fname[ ],* int *Ny,* int *Nx,* int *Npad,* double $*$ *Image*)

Reads the image stored in a FITS file and, optionally, pads it with zeros.

Reads the image stored in the FITS file 'fname', with known dimensions Nx and Ny [to be obtained using readFIT↩ Sdim()]. If Npad is larger than Nx or Ny, then it pads the image so that the corresponding dimension has Npad grid points.

It returns zero if everything was OK or the FITS error code (and prints an error message) if it wasn't.

Parameters

fname[ ]	a string with the filename to be read
Nx	an int with the dimension of the "x-axis"
Ny	an int with the dimension of the "y-axis"
Npad	an int with the dimension along each direction of the padded image

Author

Dimitrios Psaltis

Version

1.0

Date

October 2, 2017

Bug No known bugs

Warning

No known warnings

Todo nothing left

7.14.2.5 int readFITSImagedim (char *fname[ ],* int ∗ *Ny,* int ∗ *Nx,* double ∗ *yScale,* double ∗ *xScale*)

Reads the image dimensions of a FITS file, both in terms of pixels and in terms of physical units.

Reads the image dimensions of the FITS file given in the argument fname. It returns the dimensions of the image (in pixels) along its two axes in Nx and Ny and the size of each pixel along the two axis in xScale and yScale.

In this and in other subroutines, the x-axis is the same as the columns and the y-axis is the same as the rows of the image. These two notations will be used interchangeably.

It returns zero if everything was OK or the FITS error code (and prints an error message) if it wasn't.

Parameters

fname[ ]	a string with the filename to be read
∗*Ny*	on return, an int pointer with the dimension of the "y-axis" (# of rows)
∗*Nx*	on return, an int pointer with the dimension of the "x-axis" (# of columns)
∗*yScale*	on return, a double pointer with the physical size of a pixel along the y-axis
∗*xScale*	on return, a double pointer with the physical size of a pixel along the x-axis

Author

Dimitrios Psaltis

Version

1.0

Date

October 2, 2017

Bug No known bugs

Warning

No known warnings

Todo nothing left

7.14.2.6 int writeFITSVis (char *fname[ ]*, int *Ny*, int *Nx*, double * *Vp*, double * *Va*, double *vScale*, double *uScale*, char *hist[ ]*)

Writes visibility amplitudes and phases into a FITS file.

Given two real arrays Va[] and Vp[] of dimensions Nx by Ny, it stores them in the FITS file 'fname'

It returns zero if everything was OK or the FITS error code (and prints an error message) if it wasn't.

Parameters

fname[ ]	a string with the filename to be read
Nx	an int with the dimension of the "x-axis"
Ny	an int with the dimension of the "y-axis"
Va[ ]	is a Nx by Ny double array with the visibility amplitudes
Vp[ ]	is a Nx by Ny double array with the visibility phases (in rad)
uScale	is a double with the physical size of each pixel in the x-direction
vScale	is a double with the physical size of each pixel in the x-direction
hist[ ]	is a string of characters to be put in the "history" field of the FITS file

Author

Dimitrios Psaltis

Version

1.0

Date

November 3, 2017

Bug No known bugs

Warning

No known warnings

Todo nothing left

7.15 math.c File Reference

```
#include "definitions.h"
#include "global.h"
```

Macros

- #define ACC 40.0

  From numerical recipes.
- #define BIGNO 1.0e10

  a very big number; from numerical recipes
- #define BIGNI 1.0e-10

  a very small number; from numerical recipes

Functions

- double bessj0 (double x)

  Returns the Bessel function J_0(x) for any real x.
- double bessj1 (double x)

  Returns the Bessel function J_1(x) for any real x.
- double bessj (int n, double x)

  Returns the Bessel function J_n(x) for any real x.
- double polarAngle (double x, double y)

  Given two Cartesian coordinates (x,y), it returns the polar angle phi of the corresponding position vector, in the range 0->2∗pi.

7.15.1 Function Documentation

7.15.1.1 double bessj (int *n*, double *x*)

Returns the Bessel function J_n(x) for any real x.

(DP) Changed all floats to doubles; Also calls the relevant Bessel functions for n=0 and n=1

Author

Numerical Recipes in C, 2nd edition, pg. 232

Version

1.0

Date

February 8, 2016

Parameters

n	an integer argument with the order of the Bessel function
x	a double argument for the Bessel function

Returns

Returns the value of the bessel function

7.15.1.2 double bessj0 (double *x*)

Returns the Bessel function J_0(x) for any real x.

(DP) Changed all floats to doubles

Author

Numerical Recipes in C, 2nd edition, pg. 232

Version

1.0

Date

February 8, 2016

Parameters

x	a double argument for the Bessel function

Returns

Returns the value of the bessel function

7.15.1.3 double bessj1 (double *x*)

Returns the Bessel function J_1(x) for any real x.

(DP) Changed all floats to doubles

Author

Numerical Recipes in C, 2nd edition, pg. 232

Version

1.0

Date

February 8, 2016

Parameters

x	a double argument for the Bessel function

Returns

Returns the value of the bessel function

7.15.1.4 double polarAngle (double *x*, double *y*)

Given two Cartesian coordinates (x,y), it returns the polar angle phi of the corresponding position vector, in the range 0->2*pi.

Useful in converting from Cartesian to Polar Coordinates

Author

Dimitrios Psaltis

Version

1.0

Date

February 19, 2016

Parameters

x	a double argument for the horizontal Cartesian coordinate
y	a double argument for the vertical Cartesian coordinate

Returns

Returns the value of the phi angle in radians

7.16 modelsImage.c File Reference

Analytic models for images.

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
```

Macros

- #define MAXCHAR 80

  maximum number of characters for strings
- #define MAXPARAM 20

  maximum number of model parameters
- #define NMODELS 2

  number of analytic models

Functions

- int imageModelCheck (char ∗model, int ∗modelNumber)

  Checks for validity of input model.
- int imageModelCheck (int modelNumber, char ∗paramstring, double param[ ])

  Checks for validity of input model.

Variables

- char modelNames [NMODELS][MAXCHAR] ={"gauss","crescent"}

  names of known analytic models
- int modelsNParam [NMODELS] ={5,5}

  number of required parameters for each component of each model

7.16.1 Detailed Description

Analytic models for images.

A set of functions to calculate the image brightness of a number of analytic models.

In this version, it contains 2 analytic models:

- gauss

A multi-gaussian model.

$$I(x,y)=\sum_{i=1}^{N}\frac{F_i}{\sqrt{2\pi\sigma_{x,i}^2\sigma_{y,i}^2}}\exp\left[-\left(\frac{x-x_{0.i}}{2\sigma_{x,i}}\right)^2-\left(\frac{y-y_{0,i}}{2\sigma_{y,i}}\right)^2\right]$$

The first model parameter is the number of Gaussian components.

The second to sixth model paramerts are the total flux F_i, the coordinates $x_{0,i}$ and $y_{0,1}$ of its center, and the dispersions $\sigma_{x,i}$ and $\sigma_{y,i}$ along the two axis for the first component.

If more than one components are present, the same list of parameters is repeated for each model.

- crescent

The crescent model of `Kmaruddin & Dexter`, 2013, MNRAS 414, 765.

Author

Dimitrios Psaltis

Version

1.0

Date

November 14, 2017

Bug No known bugs

Warning

No known warnings

Todo Nothing left

7.16.2 Function Documentation

7.16.2.1 int imageModelCheck (char ∗ *model,* int ∗ *modelNumber*)

Checks for validity of input model.

Given a model name (in "model") it checks to see if the model is valid.

For future reference, it returns an integer modelNumber for that model.

The number of known models is stored in the constant NMODELS. The names of the models are stored in the global array modelNames[NMODELS]. The integer that corresponds to a particular model is the index of that model name in the array modelNames;

Author

Dimitrios Psaltis

Version

1.0

Date

November 14, 2017

Parameters

model	a string with the model name
modelNumber	an int that, on returrn, gives the integer model number

Returns

0 if everything was ok; 1 if there was a problem

7.16.2.2 int imageModelCheck (int *modelNumber,* char ∗ *paramstring,* double *param[ ]*)

Checks for validity of input model.

Given a modelnumber (in "modelNumber") and a string of parameters (in "paramstring"), it checks to see if the number and type of parameters are valid.

Normally, the modelNumber is first identified from a string with the model name in imageModelCheck();

For future reference, it returns a double array of the parameters in param[]

Author

Dimitrios Psaltis

Version

1.0

Date

November 14, 2017

Parameters

modelNumber	an int with the number of the model to be used
param[ ]	a double array with the parameters of the model

Returns

0 if everything was ok; 1 if there was a problem

7.17 OLD/basis.c File Reference

```
#include "definitions.h"
#include "global.h"
```

Functions

- int readUVmodel (char filename[ ], int Nu, int Nv, double Uco[ ], double Vco[ ], double VisAmp[NuMax][Nv←Max], double VisPhase[NuMax][NvMax])

  Reads visibilities on the u-v plane from the external file "filename".
- int makekGrid (int Nk, double kmin, double kmax, double kco[ ])

Generates a grid of k-points.

- int uvToPolar (int Nu, int Nv, int Nk, int Nm, double Uco[ ], double Vco[ ], double VisAmp[NuMax][NvMax], double VisPhase[NuMax][NvMax], double kco[ ], double RePk[NkMax][NmMax], double ImPk[NkMax][Nm↩ Max])

  Converts a u-v map to a polar k-m map.
- int makeImageGrid (int Na, int Nb, double amax, double bmax, double aco[ ], double bco[ ])

  Generates a grid of coordinates along the image plane.
- int polarToImage (int Nk, int Nm, int Na, int Nb, int kmin, int kmax, int mmin, int mmax, double kco[ ], double RePk[NkMax][NmMax], double ImPk[NkMax][NmMax], double aco[ ], double bco[ ], double Image[NAM↩ AX][NBMAX])

  Converts a polar k-m map to an image.
- int makeUVGrid (int Nu, int Nv, double umax, double vmax, double Uco[ ], double Vco[ ])

  Generates a grid of coordinates in the u-v plane.
- int makeUVmodel (int iModel, double param[ ], int Nu, int Nv, double Uco[ ], double Vco[ ], double Vis↩ Amp[NuMax][NvMax], double VisPhase[NuMax][NvMax])

  Makes a u-v map from an analytic model.
- double polarAngle (double x, double y)

  Given two Cartesian coordinates (x,y), it returns the polar angle phi of the corresponding position vector, in the range 0->2∗pi.
- int main (void)

  main

7.17.1 Function Documentation

7.17.1.1 int main (void)

main

This is the main function from where the algorithm starts.

Author

Dimitrios Psaltis

Version

1.0

Date

January 16, 2016

7.17.1.2 int makeImageGrid (int *Na*, int *Nb*, double *amax*, double *bmax*, double *aco[ ]*, double *bco[ ]*)

Generates a grid of coordinates along the image plane.

It generates an equidistant grid of Na alpha-points and of Nb beta-points and stores the two grids in the double arrays aco[] and bco[]. The grid are from -amax -> amax and from -bmax ->bmax. Both Na and Nb need to be odd. The routine checks whether the number of grid points is less than the maximum numbers NAMAX and NBMAX stored in the global definitions.

Author

Dimitrios Psaltis

Version

1.0

Date

February 19, 2016

Parameters

Na	an int with the number of grid points in the alpha-direction
Nb	an int with the number of grid points in the beta-direction
amax	a double with the maximum value of the alpha coordinate
bmax	a double with the maximum value of the beta coordinate
aco[ ]	a double array where the alpha coordinates will be stored
bco[ ]	a double array where the beta coordinates will be stored

Returns

0 if it was successful, 1 if there was an error, e.g., too many grid points or not an odd number

7.17.1.3 int makekGrid (int *Nk*, double *kmin*, double *kmax*, double *kco[ ]*)

Generates a grid of k-points.

It generates an equidistant grid of Nk k-points and stores it in kco[]. The first grid point is at kmin and the last one is at kmax. It checks whether the number of grid points is less than the maximum Nkmax stored in the global definitions.

Author

Dimitrios Psaltis

Version

1.0

Date

February 12, 2016

Parameters

Nk	an int with the number of grid points in the k-direction
kmin	a double with the k-value of the first grid point
kmax	a double with the k-value of the last grid point
kco[ ]	a double array where the k-coordinates will be stored

Returns

0 if it was successful, 1 if there was an error, e.g., too many grid points

7.17.1.4 int makeUVGrid (int *Nu,* int *Nv,* double *umax,* double *vmax,* double *Uco[ ],* double *Vco[ ]*)

Generates a grid of coordinates in the u-v plane.

It generates an equidistant grid of Nu by Nv grid points in the u-v plane and stores the two grids in the double arrays Uco[] and Vco[]. The grid are from -umax -> umax and from -vmax ->vmax. Both Nu and Nv need to be odd. The routine checks whether the number of grid points is less than the maximum numbers NuMax and NvMax stored in the global definitions.

Author

Dimitrios Psaltis

Version

1.0

Date

February 19, 2016

Parameters

Nu	an int with the number of grid points in the u-direction
Nv	an int with the number of grid points in the v-direction
umax	a double with the maximum value of the u coordinate
vmax	a double with the maximum value of the v coordinate
Uco[ ]	a double array where the u-coordinates will be stored
Vco[ ]	a double array where the v-coordinates will be stored

Returns

0 if it was successful, 1 if there was an error, e.g., too many grid points or not an odd number

7.17.1.5 int makeUVmodel (int *iModel,* double *param[ ],* int *Nu,* int *Nv,* double *Uco[ ],* double *Vco[ ],* double *VisAmp[NuMax][NvMax],* double *VisPhase[NuMax][NvMax]*)

Makes a u-v map from an analytic model.

On return all columns are stored in the relevant arrays in the argument list.

Author

Dimitrios Psaltis

Version

1.0

Date

February 21, 2016

Parameters

iModel	an int that specifies which model to use
param[ ]	a double array with the model parameters
Nu	an int with the number of grid points in the u-direction
Nv	an int with the number of grid points in the v-direction
Uco[ ]	a double array with the u-coordinates
Vco[ ]	a double array with the v-coordinates
VisAm[ ][ ]	a double array in which the visibility amplitudes are stored on return
VisPhase[ ][ ]	a double array in which the visibility phases are stored on return

Returns

0 if it was successful, 1 if there was an error

Todo add more models; for now it only has the crescent model

7.17.1.6 double polarAngle (double *x*, double *y*)

Given two Cartesian coordinates (x,y), it returns the polar angle phi of the corresponding position vector, in the range 0->2∗pi.

Useful in converting from Cartesian to Polar Coordinates

Author

Dimitrios Psaltis

Version

1.0

Date

February 19, 2016

Parameters

x	a double argument for the horizontal Cartesian coordinate
y	a double argument for the vertical Cartesian coordinate

Returns

Returns the value of the phi angle in radians

7.17.1.7 int polarToImage (int *Nk*, int *Nm*, int *Na*, int *Nb*, int *kmin*, int *kmax*, int *mmin*, int *mmax*, double *kco[ ]*, double *RePk[NkMax][NmMax]*, double *ImPk[NkMax][NmMax]*, double *aco[ ]*, double *bco[ ]*, double *Image[NAMAX][NBMAX]*)

Converts a polar k-m map to an image.

Given a grid of (Nk,Nm) points of the polar transform (with the real part stored in RePk[][] and the imaginary part in ImPk[][]), it returns the corresponding image on a (Na,Nb) grid of points, with the grid of alpha and beta points stored in aco[] and bco[] and the image stored in Image[][].

It will only use the k-modes between kmin and kmax as well as the m-modes between mmin and mmax, as given in the inputs

It uses the trapezoid rule to perform the integral over k and a simple summation for m. NB: it assumes that the k-coordinates are equidistant.

Author

Dimitrios Psaltis

Version

1.0

Date

February 19, 2016

Parameters

Nk	an int with the number of grid points in the k-direction
Nm	an int with the number of grid points in the m-direction
Na	an int with the number of grid points in the alpha-direction
Nb	an int with the number of grid points in the beta-direction
kmin	an int with the minimum k-mode to be used
kmax	an int with the maximum k-mode to be used
mmin	an int with the minimum m-mode to be used
mmax	an int with the maximum m-mode to be used
kco[ ]	a double array with the k-coordinates
RePk[ ][ ]	a double array in which the real parts of the polar transform will be stored
ImPk[ ][ ]	a double array in which the imaginary parts of the polar transform will be stored
aco[ ]	a double array with the alpha-coordinates
bco[ ]	a double array with the beta-coordinates
Image[ ][ ]	a double array with the image brightness

Returns

0 if it was successful, 1 if there was an error

7.17.1.8 int readUVmodel (char *filename[ ],* int *Nu,* int *Nv,* double *Uco[ ],* double *Vco[ ],* double *VisAmp[NuMax][NvMax],* double *VisPhase[NuMax][NvMax]*)

Reads visibilities on the u-v plane from the external file "filename".

The file is assumed to have 4 ASCII columns, separated by white spaces: U coordinate, V coordinate, Visibility Amplitude, Visibility Phase. The rows are assumed to be organized in such a way that for each value of the U coordinate, all V coordinates are listed sequentially. The grid is assumed to be equidistant along both directions, and this is checked during the reading. On return all columns are stored in the relevant arrays in the argument list.

Author

Dimitrios Psaltis

Version

1.0

Date

January 16, 2016

Parameters

filename[ ]	a char array with the name of the file to be read
Nu	an int with the number of grid points in the u-direction
Nv	an int with the number of grid points in the v-direction
Uco[ ]	a double array in which the u-coordinates are stored on return
Vco[ ]	a double array in which the v-coordinates are stored on return
VisAm[ ][ ]	a double array in which the visibility amplitudes are stored on return
VisPhase[ ][ ]	a double array in which the visibility phases are stored on return

Returns

0 if it was successful, 1 if there was an error in reading the file

7.17.1.9 int uvToPolar (int *Nu,* int *Nv,* int *Nk,* int *Nm,* double *Uco[ ],* double *Vco[ ],* double *VisAmp[NuMax][NvMax],* double *VisPhase[NuMax][NvMax],* double *kco[ ],* double *RePk[NkMax][NmMax],* double *ImPk[NkMax][NmMax]*)

Converts a u-v map to a polar k-m map.

Given a grid of (Nu,Nv) u-v visibility amplituded (stored in VisAmp[][]) and phases (stored in VisPhas[][]), it return the polar transform with (Nk,Nm) k-m points, with the real part of the polar transform stored in RePk[][] and the imaginary part in ImPk[][]. The grid of k-points is in kco[].

Author

Dimitrios Psaltis

Version

1.0

Date

February 12, 2016

Parameters

Nu	an int with the number of grid points in the u-direction
Nv	an int with the number of grid points in the v-direction
Nk	an int with the number of grid points in the k-direction
Nm	an int with the number of grid points in the m-direction
Uco[ ]	a double array with the u-coordinates
Vco[ ]	a double array with the v-coordinates
VisAm[ ][ ]	a double array with the visibility amplitudes
VisPhase[ ][ ]	a double array with the visibility phases
kco[ ]	a double array with the k-coordinates
RePk[ ][ ]	a double array in which the real parts of the polar transform will be stored
ImPk[ ][ ]	a double array in which the imaginary parts of the polar transform will be stored

Returns

0 if it was successful, 1 if there was an error

Todo It performs the phi integration given NPHI=128 points. In principle, this should depend on k. Given that the u-v grid is Cartesian, NPHI should be small at low k and large at large k.

7.18 OLD/beam.c File Reference

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <unistd.h>
```

Macros

- #define Ntotal 6217

    total number of data
- #define NX 255

    number of grid points on image plane along x-axis (needs to be odd)
- #define NY 255

number of grid points on image plane along y-axis (needs to be odd)
- #define FRACTION 5.0

  image size=FRACTION/minimum baseline
- #define DEFAULTOUTFILENAME "image.out"

  default filename for output, if -o option is not given

Functions

- void readdata (void)

  Reads uv tracks from filename.
- void printhelp (void)

  Prints a help message when the -h option is given.
- int parse (int argc, char ∗argv[ ])

  Parses the command line for options.
- int main (int argc, char ∗argv[ ])

  Main.

Variables

- char filename [12] ="uv60sec.dat"

  file name with data
- char ∗ outFileName

  file name with data
- char ∗ inFileName

  file name with data
- double Uco [Ntotal]

  array with u positions
- double Vco [Ntotal]

  array with v positions
- double maxB =0

  max abs values of baseline in u&v
- double minB =1e20

  min abs values of baseline in u&v
- double Xco [NX]

  coordinates along x-axis
- double Yco [NY]

  coordinates along y-axis
- double Image [NX][NY]

  brightness on image plane

7.18.1 Function Documentation

7.18.1.1 int main (int *argc,* char ∗ *argv[ ]*)

Main.

No details at this point

Author

Dimitrios Psaltis

Version

1.0

Date

September 6, 2015

Precondition

Nothing

7.18.1.2 int parse (int *argc,* char ∗ *argv[ ]*)

Parses the command line for options.

Author

Dimitrios Psaltis

Version

1.0

Date

September 7, 2015

Precondition

It is called from the main routine

Parameters

7.18.1.3 void printhelp (void)

Prints a help message when the -h option is given.

Author

Dimitrios Psaltis

Version

1.0

Date

September 7, 2015

Precondition

It is called from parse()

Parameters

no	parameters

Returns

nothing

7.18.1.4 void readdata (void)

Reads uv tracks from filename.

No details at this point

Author

Dimitrios Psaltis

Version

1.0

Date

September 6, 2015

Precondition

Nothing

Parameters

no	parameters

Returns

Returns nothing

7.19 OLD/definitions.h File Reference

Macros

- #define SmallNumber 1.e-4

  A small number to check for grid equidistance.
- #define NuMax 512

  Maximum number of grid points in u-direction (for u-v planes)
- #define NvMax 512

  Maximum number of grid points in v-direction (for u-v planes)
- #define NkMax 512

  Maximum number of grid points in k-direction (for polar transforms)
- #define NmMax 128

  Maximum number of grid points in phi-direction (for polar transforms)
- #define NAMAX 129

  Maximum number of grid points in alpha-direction (for images)
- #define NBMAX 129

  Maximum number of grid points in alpha-direction (for images)
- #define NPARMAX 8

  Maximum number of model parameters.
- #define ERROR_BOUNDS -99999.0

  Error code to be returned when extrapolating out of array bounds.

7.20 OLD/global.h File Reference

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
```

7.21 OLD/modelsUV.c File Reference

```
#include "definitions.h"
#include "global.h"
```

Functions

- double bessj1 (double x)

  Returns the Bessel function J_1(x) for any real x.
- double polarAngle (double x, double y)

  Given two Cartesian coordinates (x,y), it returns the polar angle phi of the corresponding position vector, in the range 0->2∗pi.
- int makeUVGrid (int Nu, int Nv, double umax, double vmax, double Uco[ ], double Vco[ ])

  Generates a grid of coordinates in the u-v plane.
- int Vcres (double u, double v, double R, double psi, double tau, double phi, double ∗Vreal, double ∗Vimg)

  The visibility function of a crescent model.
- int makeUVmodel (int iModel, double param[ ], int Nu, int Nv, double Uco[ ], double Vco[ ], double Vis↩Amp[NuMax][NvMax], double VisPhase[NuMax][NvMax])

  Makes a u-v map from an analytic model.

7.21.1 Function Documentation

7.21.1.1 double bessj1 (double *x*)

Returns the Bessel function J_1(x) for any real x.

(DP) Changed all floats to doubles

Author

Numerical Recipes in C, 2nd edition, pg. 232

Version

1.0

Date

February 8, 2016

Parameters

x	a double argument for the Bessel function

Returns

Returns the value of the bessel function

7.21.1.2 int makeUVGrid (int *Nu,* int *Nv,* double *umax,* double *vmax,* double *Uco[ ],* double *Vco[ ]*)

Generates a grid of coordinates in the u-v plane.

It generates an equidistant grid of Nu by Nv grid points in the u-v plane and stores the two grids in the double arrays Uco[] and Vco[]. The grid are from -umax -> umax and from -vmax ->vmax. Both Nu and Nv need to be odd. The routine checks whether the number of grid points is less than the maximum numbers NuMax and NvMax stored in the global definitions.

Author

Dimitrios Psaltis

Version

1.0

Date

February 19, 2016

Parameters

Nu	an int with the number of grid points in the u-direction
Nv	an int with the number of grid points in the v-direction
umax	a double with the maximum value of the u coordinate
vmax	a double with the maximum value of the v coordinate
Uco[ ]	a double array where the u-coordinates will be stored
Vco[ ]	a double array where the v-coordinates will be stored

Returns

0 if it was successful, 1 if there was an error, e.g., too many grid points or not an odd number

7.21.1.3 int makeUVmodel (int *iModel,* double *param[ ],* int *Nu,* int *Nv,* double *Uco[ ],* double *Vco[ ],* double *VisAmp[NuMax][NvMax],* double *VisPhase[NuMax][NvMax]*)

Makes a u-v map from an analytic model.

On return all columns are stored in the relevant arrays in the argument list.

Author

Dimitrios Psaltis

Version

1.0

Date

February 21, 2016

Parameters

iModel	an int that specifies which model to use
param[ ]	a double array with the model parameters
Nu	an int with the number of grid points in the u-direction
Nv	an int with the number of grid points in the v-direction
Uco[ ]	a double array with the u-coordinates
Vco[ ]	a double array with the v-coordinates
VisAm[ ][ ]	a double array in which the visibility amplitudes are stored on return
VisPhase[ ][ ]	a double array in which the visibility phases are stored on return

Returns

0 if it was successful, 1 if there was an error

Todo add more models; for now it only has the crescent model

7.21.1.4 double polarAngle (double *x*, double *y*)

Given two Cartesian coordinates (x,y), it returns the polar angle phi of the corresponding position vector, in the range 0->2∗pi.

Useful in converting from Cartesian to Polar Coordinates

Author

Dimitrios Psaltis

Version

1.0

Date

February 19, 2016

Parameters

x	a double argument for the horizontal Cartesian coordinate
y	a double argument for the vertical Cartesian coordinate

Returns

Returns the value of the phi angle in radians

7.21.1.5 int Vcres (double *u*, double *v*, double *R*, double *psi*, double *tau*, double *phi*, double ∗ *Vreal*, double ∗ *Vimg*)

The visibility function of a crescent model.

Returns the real and imaginary parts of the visibility function of a crescent, following the decomposition of Kamruddin & Dexter (2013), as a difference between two offset disks. Here R=Rp is the overall radius of the crescent, psi=1-↩ Rn/Rp is the relatively thickness of the crescent, tau=1-sqrt(a^2+b^2)/(Rp-Rn) is the degree of symmetry of the crescent, and phi=atan(b/a) is the orientation of the crescent. The overall normalization is provided in the calling function.

The pair (u,v) gives the location in the u-v plane.

On return, Vreal and Vimag are the values of the real and imaginary parts

Author

Dimitrios Psaltis

Version

1.0

Date

December 19, 2015

Parameters

u	a double with the u-coordinate
v	a double with the v-coordinate
R	a double with the overall radius of the crescent
psi	a double with the relativel thickness of the crescent
tau	a double with the degree of symmetry of the crescent
phi	a double with the orientation of the crescent (in rad)
Vreal	a pointer to a double that, on return, has the real part of the visibility
Vimg	a pointer to a double that, on return, has the imagin. part of the visibility

Returns

0 if everything was ok; for now, it returns nothing else

7.22 OLD/polar.c File Reference

```
#include "definitions.h"
#include "global.h"
```

Macros

- #define **NPHI** 128

Functions

- double interpUVamp (double u, double v, int Nu, int Nv, double Uco[ ], double Vco[ ], double VisAmp[Nu↩ Max][NvMax])

  Returns the visibility amplitude at a given (u,v) point by interpolating the input grid.
- double interpUVphase (double u, double v, int Nu, int Nv, double Uco[ ], double Vco[ ], double VisPhase[Nu↩ Max][NvMax])

  Returns the visibility phase at a given (u,v) point by interpolating the input grid.
- int makekGrid (int Nk, double kmin, double kmax, double kco[ ])

  Generates a grid of k-points.
- int uvToPolar (int Nu, int Nv, int Nk, int Nm, double Uco[ ], double Vco[ ], double VisAmp[NuMax][NvMax], double VisPhase[NuMax][NvMax], double kco[ ], double RePk[NkMax][NmMax], double ImPk[NkMax][Nm↩ Max])

  Converts a u-v map to a polar k-m map.

7.22.1 Function Documentation

7.22.1.1 double interpUVamp (double *u*, double *v*, int *Nu*, int *Nv*, double *Uco[ ]*, double *Vco[ ]*, double *VisAmp[NuMax][NvMax]*)

Returns the visibility amplitude at a given (u,v) point by interpolating the input grid.

Given an (Nu,Nv) grid of visibility amplitudes VisAmp[][] with coordinates stored in Uco[] and Vco[], the routine performs a linear interpolation using the nearest neigbohrs to calculate the visibility amplitude at an arbitrary point. The (u,v) grid is assumed to be equidistant in both directions.

Author

Dimitrios Psaltis

Version

1.0

Date

February 12, 2016

Parameters

u	a double with the u coordinate for which the amplitude is needed
v	a double with the v coordinate for which the amplitude is needed
Nu	an int with the number of grid points in the u-direction
Nv	an int with the number of grid points in the v-direction
Uco[ ]	a double array with the u-coordinates
Vco[ ]	a double array with the v-coordinates
VisAmp[ ][ ]	a double array with the visibility amplitudes

Returns

the visibility amplitude at the input (u,v) coordinates; it returns error code -99999 if (u,v) is out of bounds

7.22.1.2 double interpUVphase (double *u*, double *v*, int *Nu*, int *Nv*, double *Uco[ ]*, double *Vco[ ]*, double *VisPhase[NuMax][NvMax]*)

Returns the visibility phase at a given (u,v) point by interpolating the input grid.

Given an (Nu,Nv) grid of visibility amplitudes VisPhase[][] with coordinates stored in Uco[] and Vco[], the routine performs a linear interpolation using the nearest neigbohrs to calculate the visibility phase at an arbitrary point. The (u,v) grid is assumed to be equidistant in both directions.

Author

Dimitrios Psaltis

Version

1.0

Date

February 12, 2016

Parameters

u	a double with the u coordinate for which the amplitude is needed
v	a double with the v coordinate for which the amplitude is needed
Nu	an int with the number of grid points in the u-direction
Nv	an int with the number of grid points in the v-direction
Uco[ ]	a double array with the u-coordinates
Vco[ ]	a double array with the v-coordinates
VisPhase[ ][ ]	a double array with the visibility amplitudes

Returns

the visibility amplitude at the input (u,v) coordinates; it returns error code ERROR_BOUNDS if (u,v) is out of bounds

7.22.1.3 int makekGrid (int *Nk*, double *kmin*, double *kmax*, double *kco[ ]*)

Generates a grid of k-points.

It generates an equidistant grid of Nk k-points and stores it in kco[]. The first grid point is at kmin and the last one is at kmax. It checks whether the number of grid points is less than the maximum Nkmax stored in the global definitions.

Author

Dimitrios Psaltis

Version

1.0

Date

February 12, 2016

Parameters

Nk	an int with the number of grid points in the k-direction
kmin	a double with the k-value of the first grid point
kmax	a double with the k-value of the last grid point
kco[ ]	a double array where the k-coordinates will be stored

Returns

0 if it was successful, 1 if there was an error, e.g., too many grid points

7.22.1.4 int uvToPolar (int *Nu,* int *Nv,* int *Nk,* int *Nm,* double *Uco[ ],* double *Vco[ ],* double *VisAmp[NuMax][NvMax],* double *VisPhase[NuMax][NvMax],* double *kco[ ],* double *RePk[NkMax][NmMax],* double *ImPk[NkMax][NmMax]*)

Converts a u-v map to a polar k-m map.

Given a grid of (Nu,Nv) u-v visibility amplituded (stored in VisAmp[][]) and phases (stored in VisPhas[][]), it return the polar transform with (Nk,Nm) k-m points, with the real part of the polar transform stored in RePk[][] and the imaginary part in ImPk[][]. The grid of k-points is in kco[].

Author

Dimitrios Psaltis

Version

1.0

Date

February 12, 2016

Parameters

Nu	an int with the number of grid points in the u-direction
Nv	an int with the number of grid points in the v-direction
Nk	an int with the number of grid points in the k-direction
Nm	an int with the number of grid points in the m-direction

Parameters

Uco[ ]	a double array with the u-coordinates
Vco[ ]	a double array with the v-coordinates
VisAm[ ][ ]	a double array with the visibility amplitudes
VisPhase[ ][ ]	a double array with the visibility phases
kco[ ]	a double array with the k-coordinates
RePk[ ][ ]	a double array in which the real parts of the polar transform will be stored
ImPk[ ][ ]	a double array in which the imaginary parts of the polar transform will be stored

Returns

0 if it was successful, 1 if there was an error

Todo It performs the phi integration given NPHI=128 points. In principle, this should depend on k. Given that the u-v grid is Cartesian, NPHI should be small at low k and large at large k.

7.23 OLD/readUV.c File Reference

```
#include "definitions.h"
#include "global.h"
```

Functions

- int readUVmodel (char filename[ ], int Nu, int Nv, double Uco[ ], double Vco[ ], double VisAmp[NuMax][Nv↵Max], double VisPhase[NuMax][NvMax])

  Reads visibilities on the u-v plane from the external file "filename".
- double interpUVamp (double u, double v, int Nu, int Nv, double Uco[ ], double Vco[ ], double VisAmp[Nu↵Max][NvMax])

  Returns the visibility amplitude at a given (u,v) point by interpolating the input grid.
- double interpUVphase (double u, double v, int Nu, int Nv, double Uco[ ], double Vco[ ], double VisPhase[Nu↵Max][NvMax])

  Returns the visibility phase at a given (u,v) point by interpolating the input grid.

7.23.1 Function Documentation

7.23.1.1 double interpUVamp (double *u*, double *v*, int *Nu*, int *Nv*, double *Uco[ ]*, double *Vco[ ]*, double *VisAmp[NuMax][NvMax]*)

Returns the visibility amplitude at a given (u,v) point by interpolating the input grid.

Given an (Nu,Nv) grid of visibility amplitudes VisAmp[][] with coordinates stored in Uco[] and Vco[], the routine performs a linear interpolation using the nearest neigbohrs to calculate the visibility amplitude at an arbitrary point. The (u,v) grid is assumed to be equidistant in both directions.

Author

Dimitrios Psaltis

Version

1.0

Date

February 12, 2016

Parameters

u	a double with the u coordinate for which the amplitude is needed
v	a double with the v coordinate for which the amplitude is needed
Nu	an int with the number of grid points in the u-direction
Nv	an int with the number of grid points in the v-direction
Uco[ ]	a double array with the u-coordinates
Vco[ ]	a double array with the v-coordinates
VisAmp[ ][ ]	a double array with the visibility amplitudes

Returns

the visibility amplitude at the input (u,v) coordinates; it returns error code -99999 if (u,v) is out of bounds

7.23.1.2 double interpUVphase (double *u*, double *v*, int *Nu*, int *Nv*, double *Uco[ ]*, double *Vco[ ]*, double *VisPhase[NuMax][NvMax]*)

Returns the visibility phase at a given (u,v) point by interpolating the input grid.

Given an (Nu,Nv) grid of visibility amplitudes VisPhase[][] with coordinates stored in Uco[] and Vco[], the routine performs a linear interpolation using the nearest neigbohrs to calculate the visibility phase at an arbitrary point. The (u,v) grid is assumed to be equidistant in both directions.

Author

Dimitrios Psaltis

Version

1.0

Date

February 12, 2016

Parameters

u	a double with the u coordinate for which the amplitude is needed
v	a double with the v coordinate for which the amplitude is needed
Nu	an int with the number of grid points in the u-direction
Nv	an int with the number of grid points in the v-direction
Uco[ ]	a double array with the u-coordinates
Vco[ ]	a double array with the v-coordinates
VisPhase[ ][ ]	a double array with the visibility amplitudes

Returns

the visibility amplitude at the input (u,v) coordinates; it returns error code ERROR_BOUNDS if (u,v) is out of bounds

7.23.1.3 int readUVmodel (char *filename[ ],* int *Nu,* int *Nv,* double *Uco[ ],* double *Vco[ ],* double *VisAmp[NuMax][NvMax],* double *VisPhase[NuMax][NvMax]*)

Reads visibilities on the u-v plane from the external file "filename".

The file is assumed to have 4 ASCII columns, separated by white spaces: U coordinate, V coordinate, Visibility Amplitude, Visibility Phase. The rows are assumed to be organized in such a way that for each value of the U coordinate, all V coordinates are listed sequentially. The grid is assumed to be equidistant along both directions, and this is checked during the reading. On return all columns are stored in the relevant arrays in the argument list.

Author

Dimitrios Psaltis

Version

1.0

Date

January 16, 2016

Parameters

filename[ ]	a char array with the name of the file to be read
Nu	an int with the number of grid points in the u-direction
Nv	an int with the number of grid points in the v-direction
Uco[ ]	a double array in which the u-coordinates are stored on return
Vco[ ]	a double array in which the v-coordinates are stored on return
VisAm[ ][ ]	a double array in which the visibility amplitudes are stored on return
VisPhase[ ][ ]	a double array in which the visibility phases are stored on return

Returns

0 if it was successful, 1 if there was an error in reading the file

7.24 synthimage.c File Reference

Creates a synthetic static image based on a model.

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include "fitsio.h"
```

Macros

- #define VMODEDEFAULT 1

  default verbose mode "medium"
- #define MAXCHAR 80

  maximum number of characters for strings
- #define MAXPIXEL 4096

  maximum number of pixels per size of image
- #define NPIXELDEFAULT 512

  default number of pixels
- #define PIXELSIZEDEFAULT 1.0

  default pixel size
- #define MODELDEFAULT "gauss"

  default model
- #define PARAMDEFAULT "1,0.0,0.0,20.0,20.0"

  default parameter array
- #define MAXPARAM 20

  maximum number of model parameters
- #define RED "\x1B[31m"

  color RED for error output
- #define RESETCOLOR "\x1B[0m"

  color to reset to normal

Functions

- void printErrorSynthimage (char errmsg[ ])

  Prints an error message.
- void printhelp (void)

  Prints a help message when no other arguments are given.
- void verboseinput (char ∗outFileName, int ∗vmode, int ∗cmode, int ∗Npad)

  Asks for user input for all the options of this command.
- int parse (int argc, char ∗argv[ ], char ∗outFileName, int ∗vmode, int ∗Npixel, double ∗pixelSize, char ∗model, char ∗paramstring)

  Parses the command line for options.
- int main (int argc, char ∗argv[ ])

  Main program.

7.24.1 Detailed Description

Creates a synthetic static image based on a model.

This program creates a synthetic static square image from a model and stores the result in an output FITS file.

Use: synthimage [-sv] -p Npixels -c size -m modelname -d param1,param2,... filename

The required option is:

- "filename": sets the output image filename (FITS)

The optional options are:

- "-p Npixels": sets the number of image pixels per dimension (default 512)
- "-s size": physical dimension of each pixel in microarcsec (default 1.0)
- "-m modelname": the name of the model to be used (default "gauss")
- "-q param1,param2,...": the values of the various model parameters (separated by commas, with no spaces between them or in quotes) (default 1,0.0,0.0,20.0,20.0)
- "-s": silent mode. It does not print anything and uses defaults
- "-v": verbose mode. It prints a lot more information

If no options are given, it prints a help message

Examples:

- synthimage -p512 -c1.0 -m gauss -d 1,0.,0.,10.0,2.0 image.fits

Creates a synthetic image with 512 pixels along its side, with each pixel having a physical dimension of 1 microarcsec. The image is created from a gaussian model, with one gaussian component, centered at (0.0,0.0) microarcsec from the center of the image and with standard deviation equal to 10.0 and 2.0 microarcsec along the x- and y-orientations.

Author

Dimitrios Psaltis

Version

1.0

Date

November 12, 2017

Bug No known bugs

Warning

No known warnings

Todo Nothing to do

7.24.2 Function Documentation

7.24.2.1 int main (int *argc,* char ∗ *argv[ ]*)

Main program.

Author

Dimitrios Psaltis

Version

1.0

Date

September 30, 2017

Precondition

Nothing

7.24.2.2 int parse (int *argc,* char ∗ *argv[ ],* char ∗ *outFileName,* int ∗ *vmode,* int ∗ *Npixel,* double ∗ *pixelSize,* char ∗ *model,* char ∗ *paramstring*)

Parses the command line for options.

Parses the command line for options. If no options are given, it prints a help message

The required options are:

- <filename>: sets the input image filename (FITS)

The optional options are:

- "-o <filename>": sets the output visibility filename (UVFITS).
- "-s": silent mode. It does not print anything
- "-v": verbose mode. It prints a lot more information
- "-p Npoints": pading. It pads the image to a square grid with Npoints on each side
- "-c": calculates the complex phase by first centering the image to its center of brightness.

Author

Dimitrios Psaltis

Version

1.0

Date

September 30, 2017

Precondition

It is called from main()

Parameters

argc	an int (as is piped from the unix prompt)
argv[ ]	an array of strings (as is piped from the unix prompt)
∗*inFileName*	a string which returns the input filename
∗*outFileName*	a string which returns the output filename
∗*vmode*	an int with a flag for the chosen verbose mode (0:silent, 1: normal, 2: verbose)
∗*cmode*	an int with a flat for whether the FFT will be centered on the center of brightness
∗*Npad*	an int with the number of points per dimension to which the image will be padded. It it is smaller than the number of points in the image, then the total number of points in the image will be used.

Returns

Returns zero if successful, 1 if not

7.24.2.3 void printErrorSynthimage (char *errmsg[ ]*)

Prints an error message.

Author

Dimitrios Psaltis

Version

1.0

Date

November 12, 2017

Parameters

errmsg[ ]	a string with the error message to be printed

Returns

nothing

7.24.2.4 void printhelp (void)

Prints a help message when no other arguments are given.

Author

Dimitrios Psaltis

Version

1.0

Date

September 30, 2017

Precondition

It is called from parse() and from main()

Parameters

no	parameters

Returns

nothing

7.24.2.5 void verboseinput (char ∗ *outFileName,* int ∗ *vmode,* int ∗ *cmode,* int ∗ *Npad*)

Asks for user input for all the options of this command.

It asks the user to input directly all the possible options that one can give in this command. It is called from the main function if the '-v' option is given as an argument.

When each option is presented, the default value or the value that was passed as an argument is displayed. If the user hits return, then that value remains. If the user provides a valid answer, it replaces the value for that option.

Author

Dimitrios Psaltis

Version

1.0

Date

November 12, 2017

Precondition

It is called from main()

Parameters

∗*outFileName*	a string which provides and returns the output filename
∗*vmode*	an int with a flag for the chosen verbose mode (0:silent, 1: normal, 2: verbose)
∗*cmode*	an int with a flat for whether the FFT will be centered on the center of brightness
∗*Npad*	an int with the number of points per dimension to which the image will be padded. It it is smaller than the number of points in the image, then the total number of points in the image will be used.

Returns

Returns zero if successful, 1 if not

Index