1.9 synthesistester - Module

Module for testing synthesis processing
include "synthesistester.g"

Description
synthesistester contains a number of tools and functions for testing synthesis processing.

vlaendtoend

imagerpositiontest

1.9.1 imagerpositiontest - Tool

Package	synthesis
Module	synthesistester

Testing of imager using resulting source positions
include "imagerpositiontest.g"

Description

imagerpositiontest.g provides an interface for testing the capabilities of imager and pimager. It essentially creates a simulated dataset and images it, comparing the positions of the imaged sources with the positions of the known components.

Overview of `imagerpositiontest` tool functions:

cleanup done dotests findsource get_beam get_center_position get_componentlisttable get_corresponding_comps get_rms get_seed get_source_box make_default_cl make_default_simulator make_image make_model_image make_ms set_clean_mask set_clean_parms set_componentlisttable set_deconvolution_function set_imaging_data_parms set_imaging_options set_imaging_parms set_imaging_tool_type set_mem_parms set_mfcontrol_parms set_pimager_numprocs set_seed set_sim_parms set_simulator set_tolerance sourcecompare view

Tools

imagerpositiontest	tool for testing the validity of imager and pimager results

Functions

vlaendtoend	Test VLA end to end processing

1.9.2 imagerpositiontest - Tool

Package synthesis
Module synthesistester

tool for testing the validity of imager and pimager results

Description
imagerpositiontest is a tool for testing the validity of results from imager and pimager. It does this by generating a simulated dataset (so the source distribution is known), imaging that dataset, and then comparing the positions of the imaged sources with their known positions.

Constructors

imagerpositiontest	Construct an imagerpositiontest tool

Functions

cleanup	Clean up from a previous run.
done	Release the resources used by this tool.
dotests	Run the tests.
findsource	Locate source positions in an image
get_beam	Get the restoring beam plus some extra info from the restored image.
get_center_position	Get the center position for the map that will be created.
get_componentlisttable	Get the name of the disk table containing the positions of the simulated components.
get_corresponding_comps	Return the map of found components in the restored image to their true component counterparts.
get_rms	Get the rms from the restored map in an "intelligent" fashion.
get_seed	Get the seed for the random number generator
get_source_box	Get the pixel coordinates of a box encompassing the simulated sources.
make_default_cl	Make the default simulated component list.
make_default_simulator	Create the default simulator.
make_image	Make the images
make_model_image	Make an image of the simulated component distribution.
make_ms	Make the MS containing the simulated data
set_clean_mask	Specify how the clean mask should be set.
set_clean_parms	Set up the parameters for a clean test.
set_componentlisttable	Set the name of the componentlist table.
set_deconvolution_function	Set the decnvolution function to use.

set_imaging_data_parms	Set the MS parameters to be used for imaging.
set_imaging_options	Set some general options for subsequent processing.
set_imaging_parms	Set the parameters to be used for imaging.
set_imaging_tool_type	Set the imaging tool to test.
set_mem_parms	Set parameters to be used in a mem deconvolution.
set_mfcontrol_parms	Set control parameters for multi-field imaging tests.
set_pimager_numprocs	Set the number of processors available to pimager.
set_seed	Set the seed used by the random number generator for creating the simulated components.
set_sim_parms	Set up the parameters to be used when creating the default simulator.
set_simulator	Set an alternate simulator to be used for generating the MS.
set_tolerance	Set the tolerance allowed for a position test to pass.
sourcecompare	Compare the found source positions with the true source positions.
view	View the restored image.

imagerpositiontest - Constructor

Package	synthesis
Module	synthesistester
Tool	imagerpositiontest

Construct an imagerpositiontest tool

Synopsis
imagerpositiontest(testdir, clean)

Description
This is used to construct an imagerpositiontest tool. A Glish variable containing the tool functions of imagerpositiontest is returned.

Arguments

testdir	sub-directory of the current directory where imagerpositiontest will create datasets and other files Allowed: String Default: 'imager_position_test'
clean	Delete files from under testdir which have been left from a previous run Allowed: Boolean Default: T

Returns
imagerpositiontest tool

Example

```
# construct an imagerpositiontest tool
ipt := imagerpositiontest('scratchdir');
```

---

imagerpositiontest.cleanup - Function

Package	synthesis
Module	synthesistester
Tool	imagerpositiontest

Clean up from a previous run.

Synopsis
cleanup()

Description
Clean up from a previous run. This removes the contents of the directory specified during the tool creation and it is run at tool creation time if clean=T is passed to the constructor (this is the default behavior, so clean=F must be specified if you don't want this to happen).

Returns
T or fail

Example

```
- include 'imagerpositiontest.g'
# construct an imagerpositiontest tool
- ipt := imagerpositiontest('scratchdir',F);
# remove contents of scratchdir from a previous run
- ipt.cleanup();
T
```

---

imagerpositiontest.done - Function

Package synthesis
Module synthesistester
Tool imagerpositiontest

Release the resources used by this tool.

Synopsis
done()

Description
Releases the resources used by this tool by setting the private and public records to F.

Returns
T

Example

```
- include 'imagerpositiontest.g'
# construct an imagerpositiontest tool
- ipt := imagerpositiontest('scratchdir');
# oops I didn't really want to create such a tool
- ipt.done();
T
```

imagerpositiontest.dotests - Function

Package synthesis
Module synthesistester
Tool imagerpositiontest

Run the tests.

Synopsis
dotests()

Description

This is the heart of imagerpositiontest. It creates the simulator and the associated ms, creates and deconvolves the image, compares the source positions from the image to the known component positions, and summarizes these comparisons. The user may speicify numerous parameters prior to invoking dotests() via the many set_* functions, or may specify none at all, in which case sensible defaults are used.

Returns
T or fail

Example

```
# a very simple, default test
- include 'imagerpositiontest.g'
# construct an imagerpositiontest tool
- ipt := imagerpositiontest('scratchdir');
- ipt.dotests();
T
# view the restored image
- ipt.view();
- ipt.done();
```

Running dotests() using all defaults takes between 4 and 5 minutes on a Linux box with 4 500 MHz Xeon processors (only one used at a time in this implementation however) and 4 GB of RAM. Here is a more elaborate test:

```
include 'imagerpositiontest.g';
ipt := imagerpositiontest();
ok := ipt.set_sim_parms(frequency='30GHz',docorrupt=F,nsources=10,nfields=4,
                        nchan=1,scalefactor=1.2,source_spread=0.9,
                        sourcedirection=dm.direction('J2000', '17h44m0.0',
                                                     '50d0m0.000'));
ipt.set_deconvolution_function('mem');
ipt.set_mem_parms(algorithm='mfentropy',niter=20);
if(is_fail(ok)) {
    print ok::message;
    exit 1;
}
ok := ipt.dotests();
if(is_fail(ok)) {
    print ok::message;
    exit 1;
```

```
}
ipt.view();
```

---

imagerpositiontest.findsource - Function

Package	synthesis
Module	synthesistester
Tool	imagerpositiontest

Locate source positions in an image

Synopsis
findsource(numsources, imagename, clname)

Description
Locate source positions in an image. dotests() calls this function using the restored image as the target image; it is part of the user interface in case you might have a use for it. The task generates disk componentlist tables with names of the form clname.<x> where x is the channel number of the image.

Arguments

numsources	The number of sources to look for Allowed: Int > 0 Default: None
imagename	The name of the target disk image Allowed: String (any valid disk image table) Default: None
clname	The name of the generated componentlist table Allowed: String (any valid filename) Default: None

Returns
T or fail

Example

```
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
- ipt.findsource(10,'myimage.im','found_components');
T
```

---

imagerpositiontest.get_beam - Function

Package	synthesis
Module	synthesistester
Tool	imagerpositiontest

Get the restoring beam plus some extra info from the restored image.

Synopsis
get_beam()

Description
Get the restoring beam. The returned record has the following fields: major, minor, positionangle, ra, and dec. The last two are the beam FWHM in right ascension and declination. All returned fields are quantities; ie, they are records themselves having fields called value and unit.

Returns
Record or fail

Example

```
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
- ipt.dotests();
- beam := ipt.beam()
- beam;
[major=[value=0.695572555, unit=arcsec], minor=[value=0.623729467, unit=arcsec],
positionangle=[value=-11.487915, unit=deg], ra=[value=0.626736009, unit=arcsec],
dec=[value=0.69286478, unit=arcsec]]
```

---

imagerpositiontest.get_center_position - Function

Package synthesis
Module synthesistester
Tool imagerpositiontest

Get the center position for the map that will be created.

Synopsis
get_center_position()

Description
This function determines the center position for the images that will be created by getting the PHASE_DIR column from the MS's field table and averaging the max and min RA and the max and min Dec.

Returns
a Direction or fail

Example

```
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
- ipt.set_sim_parms(nfields=4,sourcedirection=dm.direction('J2000',
                                       '17h44m0.0','-10d0m0.000'));
- ipt.make_default_simulator();
- ipt.get_center_position();
[type=direction, refer=J2000, m1=[value=-0.174669194, unit=rad],
m0=[unit=rad, value=-1.64074787]]
```

---

imagerpositiontest.get_componentlisttable - Function

Package synthesis
Module synthesistester
Tool imagerpositiontest

Get the name of the disk table containing the positions of the simulated components.

Synopsis
get_componentlisttable()

Description
Get the name of the disk table containing the simulated components. Note that for this function to return a value, the componentlist table doesn't need to exist yet.

Returns
The table name as a string

Example

```
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
- ipt.get_componentlisttable();
imager_position_test/simu_comp.cl
```

---

imagerpositiontest.get_corresponding_comps - Function

Package	synthesis
Module	synthesistester
Tool	imagerpositiontest

Return the map of found components in the restored image to their true component counterparts.

Synopsis
get_corresponding_comps()

Description
Return the map of found components in the restored image to their true component counterparts. The returned array of ints is two dimensional. The first index represents the channel number and the second index represents the found component number. The value represents the corresponding simulated component number (or the simulated component which was found to be nearest the found component in the restored image).

Returns
A two dimensional array of ints

Example

```
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
- ipt.dotests();
- ipt.get_corresponding_comps();
[[1:1,]
    1 3 2 3]
# in this case one of the tests failed; the nearest simulated
# component to two of the components found in the restored map was
# simulated component 3
```

---

imagerpositiontest.get_rms - Function

Package	synthesis
Module	synthesistester
Tool	imagerpositiontest

Get the rms from the restored map in an "intelligent" fashion.

Synopsis
get_rms(im)

Description
Get the rms from the restored map in an "intelligent" fashion. The way this algorithm works is to first get the rms using the entire image and then create a mask, using pixels in the image within -3 to 3 times the rms just calculated. The algorithm then iterates in this fashion until the fractional difference between the previously calculated rms and the currently calculated rms is below some threshhold, currently hard wired to be 0.001.

Arguments

im		A valid image tool from which to get the rms
	Allowed	Any valid image tool
	Default	None

Returns
Float

Example

```
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
- include 'image.g';
- myim := image('some.image');
- rms := ipt.get_rms(myim);
# numerous messages from image.stats() here
# and finally
- rms;
0.00384885306
```

---

imagerpositiontest.get_seed - Function

Package	synthesis
Module	synthesistester
Tool	imagerpositiontest

Get the seed for the random number generator

Synopsis
get_seed()

Description
Get the seed for the random number generator. Using this seed by invoking set_seed allows one to repeat the trial just run.

Returns
Integer used to seed the random number generator

Example

```
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
```

```
- ipt.dotests();
- seed := ipt.get_seed();
- seed;
997469885
```

---

imagerpositiontest.get_source_box - Function

Package	synthesis
Module	synthesistester
Tool	imagerpositiontest

Get the pixel coordinates of a box encompassing the simulated sources.

Synopsis
get_source_box()

Description
Get the coordinates of a 2-D box encompassing the simulated sources. The pixels correspond to the restored image. A buffer zone of 5 pixels is applied to the max and min positions of the simulated components.

Returns
A record with fields blc and trc with the 2-D pixel locations.

Example

```
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
- ipt.dotests();
- box := ipt.get_source_box();
- box;
[blc=[274 280] , trc=[317 395] ]
```

---

imagerpositiontest.make_default_cl - Function

Package	synthesis
Module	synthesistester
Tool	imagerpositiontest

Make the default simulated component list.

Synopsis
make_default_cl(clname, frequency, epoch, nsources, phasecenter, fluxmin, fluxmax, minsep)

Description
Make the default simulated component list. It's usually a good idea just to let dotests() do this, but if you want some more rope, here it is....

Arguments

clname	The name of the componentlist table that will be written to disk Allowed: String (any valid file name) Default: 'dummy.cl'
frequency	The reference frequency of the component list Allowed: String (frequency expressed as a quantity) Default: unset (will use the user-specified frequency from set_sim_parms() or the tool's default frequency)
epoch	The epoch of the positions Allowed: String (any recognized epoch) Default: 'J2000'
nsources	The number of components to create Allowed: Integer > 0 Default: 4
phasecenter	The phasecenter Allowed: Direction Default: unset (uses the user specified phase center from set_sim_parms() or the tool's default phase center
fluxmin	The minimum flux a component can have in Jy Allowed: Float > 0 Default: 3
fluxmax	The maximum flux a component can have in Jy Allowed: Float => minflux Default: 10
minsep	The minimum separation between components in arcsec Allowed: Float => 0 Default: 5

Returns
T or fail

Example

```
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
- ipt.make_default_cl();
T
```

---

imagerpositiontest.make_default_simulator - Function

Package	synthesis
Module	synthesistester
Tool	imagerpositiontest

Create the default simulator.

Synopsis
make_default_simulator(frequency, arrayscale)

Description
Create the default simulator. Usually, it's a good idea just to let dotests() do this, but here is some more rope for you if you want it....

Arguments

frequency	The frequency for the output ms. Allowed: Quantity (frequency) Default: unset will use the tool default value or the user-specified value in set_sim_parms()
arrayscale	The scale (baseline lengths) of the default interferometer Allowed: Float > 0 Default: 1

Returns
T or fail

Example

```
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
- ipt.make_default_simulator();
T
```

---

imagerpositiontest.make_image - Function

Package	synthesis
Module	synthesistester
Tool	imagerpositiontest

Make the images

Synopsis
make_image(ms, imagename)

Description
Make the images. Like most make_* tasks, this is called from dotests() and most of the time you will just want to run that. This function creates and sets up the imager or pimager tool (imager by default but the user can specify via set_imaging_tool_type()). Defaults are chosen if not specified by the user via other set_* functions with deal with the configuration of the imaging tool. Unless appropriate parameters have been set by the user, the imaging tool's advise() function is used for setting some parameters. A dirty image is created, deconvolved, and restored if necessary.

Arguments

ms	The name of the MS on disk. Allowed String (any valid name of a disk MS) Default unset (will use the tool default value)
imagename	The output imagename prefix (things like .dirty and .model will be appended to label the appropriate images) Allowed String (any valid filename) Default None

Returns
T or fail

Example

```
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
- ipt.make_default_simulator();
- ipt.make_default_cl();
- ipt.make_ms();
- ipt.make_image(imagename='sim_image');
T
```

---

imagerpositiontest.make_model_image - Function

Package	synthesis
Module	synthesistester
Tool	imagerpositiontest

Make an image of the simulated component distribution.

Synopsis
make_model_image(template, imagename)

Description
Make an image of the simulated component distribution. You must specify a template image on which to put the simulated components (for ease of use, it is usually best to use an image created by make_image). This function is called by make_image() when the user runs set_clean_mask(type='comps').

Arguments

template	Name of the template disk image to use Allowed String: (any valid disk image name) Default None
imagename	Name of the output model image Allowed String (any valid file name) Default 'simulimage_orig.model'

Returns
T or fail

Example

```
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
- ipt.dotests();
- ipt.make_model_image('imager_position_test/simulimage.dirty','new_model.im');
T
- # and view it to see the simulated sky brightness
- myim := image('new_model.im');
- myim.view();
```

---

imagerpositiontest.make_ms - Function

Package	synthesis
Module	synthesistester
Tool	imagerpositiontest

Make the MS containing the simulated data

Synopsis
make_ms(clfile, docorrupt, noiselevel)

Description
Make the MS containing the simulated data. It is usually a good idea just to let dotests() call this function.

Arguments

clfile	The name of the coponentlist disk table to use for generating the MS Allowed String (a valid componentlist table name) Default unset (will use the tool's default or that specified by the user in set_componentlisttable())
docorrupt	Should noise be added to the MS data? Allowed Boolean Default F
noiselevel	If docorrupt==T, the level of the noise that is passed to simulator.corrupt() Allowed String (a valid flux density quantity) Default F

Returns
T or fail

Example

```
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
- ipt.make_default_simulator();
- ipt.make_default_cl();
# make the ms and corrupt with a noise level of 1 Jy
- ipt.make_ms('imager_position_test/simu_comp.cl',T,'1Jy');
T
```

---

imagerpositiontest.set_clean_mask - Function

Package synthesis
Module synthesistester
Tool imagerpositiontest

Specify how the clean mask should be set.

Synopsis
set_clean_mask(type)

Description

When the test is to be carried out using clean, this specifies how the clean mask should be set. There are two recognized settngs. The default, 'all' means to not use a mask when cleaning. 'comps' means to search only the area around the simulated components (a mask image corresponding to this request is created by make_image() by calling make_model_image()).

Arguments

type	The mask type to use when cleaning Allowed String: 'all'|'comps' Default: 'all'

Returns
T or fail

Example

```
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
- ipt.set_deconvolution_function('clean');
# search for clean components only in the areas around the simulated sources
- ipt.set_clean_mask('comps');
T
- ipt.dotests();
```

---

imagerpositiontest.set_clean_parms - Function

Package	synthesis
Module	synthesistester
Tool	imagerpositiontest

Set up the parameters for a clean test.

Synopsis
set_clean_parms(algorithm, niter, gain, threshold, displayprogress, fixed)

Description

Set up the parameters for a clean test. The parameters set here are passed to (p)imager.clean() from make_image().

Arguments

algorithm	The clean algorithm to use Allowed: String: 'clark'|'hogbom'|'mfclark'|'mfhogbom'|'wfclark'|'wfhogbom' Default: 'clark'
niter	Number of clean iterations Allowed: Integer >= 0 Default: 1000
gain	Loop gain factor Allowed: Float > 0 Default: 0.1
threshold	Residual level at which to stop cleaning Allowed: Quantity (flux density) Default: '0Jy'
displayprogress	Display the progress during the clean? Allowed: Boolean Default: F
fixed	Keep one or more models fixed? Allowed: Array of booleans Default: F

Returns
T or fail

Example

```
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
# set up a multifield clark clean test
- ipt.set_deconvolution_function('clean');
- ipt.set_sim_parms(nfields=4);
- ipt.set_clean_parms(algorithm='mfclark',niter=2000,gain=0.15);
T
- ipt.dotests();
```

---

imagerpositiontest.set_componentlisttable - Function

Package synthesis
Module synthesistester
Tool imagerpositiontest

Set the name of the componentlist table.

Synopsis
set_componentlisttable(cltablename)

Description
Set the name of the componentlist table to use for generating the MS to be imaged. Thus, the user can specify the model to use in this way. If not specified, dotests() will call make_default_cl().

Arguments

cltablename		The disk componentlist table name
	Allowed	String (any valid disk component list table name)
	Default	None

Returns
T

Example

```
# do a test using a user-created component list
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
- ipt.set_componentlisttable('mymodel.cl');
T
# set up a multifield clark clean test
- ipt.set_deconvolution_function('clean');
- ipt.set_sim_parms(nfields=4);
- ipt.set_clean_parms(algorithm='mfclark',niter=2000,gain=0.15);
- ipt.dotests();
```

---

imagerpositiontest.set_deconvolution_function - Function

Package	synthesis
Module	synthesistester
Tool	imagerpositiontest

Set the decnvolution function to use.

Synopsis
set_deconvolution_function(df)

Description
Set the deconvolution function to use. Currently, 'mem' and 'clean' are recognized.

Arguments

df	The deconvolution function to use AllowedString: 'clean'|'mem' Default:'clean'

Returns
T or fail

Example

```
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
# set up a single field mem test
- ipt.set_deconvolution_function('mem');
T
- ipt.set_sim_parms(nfields=1);
- ipt.set_mem_parms(algorithm='entropy',niter=25,sigma='0.1Jy');
- ipt.dotests();
```

---

imagerpositiontest.set_imaging_data_parms - Function

Package	synthesis
Module	synthesistester
Tool	imagerpositiontest

Set the MS parameters to be used for imaging.

Synopsis

set_imaging_data_parms(mode, nchan, start, step, msselect)

Description

Set the MS parameters to be used for imaging. These parameters are passed to (p)imager.setdata() by make_image(). You sould only need to call this function when you explicitly want to set these parameters; otherwise, sensible defaults are used.

Arguments

mode	Type of processing Allowed: String: 'channel'|'velocity' Default: 'channel'
nchan	Number of channels Allowed: Integer > 0 Default: Unset (will use number of channels in the simulated MS
start	Starting channel Allowed: 0 > Integer >= number of channels in data set Default: 1
step	Step in channels Allowed: Integer > 0 Default: 1
msselect	TaQL select string applied as a logical "and" with the other selections Allowed: String (valid TaQL query) Default: Unset

Returns

T

Example

```
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
# set up a single field mem test, dropping some baselines
- ipt.set_deconvolution_function('mem');
- ipt.set_sim_parms(nfields=1);
- ipt.set_mem_parms(algorithm='entropy',niter=25,sigma='0.1Jy');
- ipt.set_imaging_data_parms(msselect='ANTENNA1 <= 20');
T
- ipt.dotests();
```

imagerpositiontest.set_imaging_options - Function

Package synthesis
Module synthesistester
Tool imagerpositiontest

Set some general options for subsequent processing.

Synopsis
set_imaging_options(ftpmachine, cache, tile, gridfunction, location, padding)

Description
Set some general options for subsequent processing. These options are passed to (p)imager.setoptions() by make_image().

Arguments

ftpmachine	Fourier transform machine Allowed: 'gridft' Default: 'gridft'
cache	Size of gridding cache in complex pixels Allowed: Integer >= 0 Default: 0
tile	Size of a gridding tile in pixels (in 1 dimension) Allowed: Integer > 0 Default: 16
gridfunction	Gridding function Allowed: String: 'SF'|'BOX' Default: 'SF'
location	Location used in phase rotations Allowed: position measure Default: F
padding	Padding factor in image plane Allowed: Float >= 1.0 Default: 1.0

Returns
T

Example

```
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
# set up a single field mem test, changing the default padding
- ipt.set_deconvolution_function('mem');
- ipt.set_sim_parms(nfields=1);
- ipt.set_mem_parms(algorithm='entropy',niter=25,sigma='0.1Jy');
- ipt.set_imaging_options(padding=1.7);
T
- ipt.dotests();
```

---

imagerpositiontest.set_imaging_parms - Function

Package	synthesis
Module	synthesistester
Tool	imagerpositiontest

Set the parameters to be used for imaging.

Synopsis
set_imaging_parms(nx, ny, cellx, celly, stokes, doshift,
phasecenter, shiftx, shifty, mode, nchan, start, step, facets)

Description
Set the parameters to be used for imaging. These parameters are passed to (p)imager.setimage() by make_image(). For the most part, you don't have to call this function unless you want to explicitly set some parameters; the correct thing will be done otherwise.

Arguments

nx	Number of pixels in the x direction. Allowed: Integer > 0 Default: Unset. (determined by (p)imager.advise())
ny	Number of pixels in the y direction. Allowed: Integer > 0 Default: Unset. (determined by (p)imager.advise())
cellx	Cell size in the x direction Allowed: Qunatity (angle) Default: Unset. (determined by (p)imager.advise())
celly	Cell size in the y direction Allowed: Qunatity (angle) Default: Unset. (determined by (p)imager.advise())
stokes	Stokes parameters to image Allowed: String: 'I'|'IV'|'IQU'|'IQUV' Default: Unset (will be set to 'I')
doshift	Use the specified phase center? Allowed: Boolean Default: Unset (will be set according to if the test is for a single field or multiple fields
phasecenter	Direction of phase center as a measure Allowed: MDirection Default: Unset (will be set if the the test is multi field)
shiftx	Shift in x Allowed: Quantity (angle) Default: '0arcsec'
shifty	Shift in y Allowed: Quantity (angle) Default: '0arcsec'
mode	Type of processing Allowed: String: 'mfs'|'channel'|'velocity' Default: 'channel'
nchan	Number of channels Allowed: Integer > 0 Default: Unset (will use number of channels in the simulated data set)
start	Starting channel Allowed: 0 > Integer >= number of channels in data set Default: 1
step	Step in channels Allowed: Integer > 0 Default: 1
facets	Number of facets on each axis Allowed: Integer > 0 Default: Unset (determined by (p)imager.advise())

Returns
T

Example

```
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
# set up a single field mem test and shift the phase center
- ipt.set_deconvolution_function('mem');
- ipt.set_sim_parms(nfields=1);
- ipt.set_mem_parms(algorithm='entropy',niter=25,sigma='0.1Jy');
- ipt.set_imaging_parms(shiftx='20arcsec',shifty='-10arcsec');
T
- ipt.dotests();
```

---

imagerpositiontest.set_imaging_tool_type - Function

Package	synthesis
Module	synthesistester
Tool	imagerpositiontest

Set the imaging tool to test.

Synopsis
set_imaging_tool_type(type)

Description
Set the imaging tool to test, 'imager' or 'pimager'.

Arguments

type	Tool to use AllowedString: 'imager'|'pimager Default:'imager'

Returns
T

Example

```
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
# set up a single field mem test using pimager
- ipt.set_imaging_tool_type('pimager');
T
- ipt.set_pimager_numprocs(16);
- ipt.set_deconvolution_function('mem');
- ipt.set_sim_parms(nfields=1,nchan=128);
- ipt.set_mem_parms(algorithm='entropy',niter=25,sigma='0.1Jy');
- ipt.set_imaging_parms(shiftx='20arcsec',shifty='-10arcsec');
- ipt.dotests();
```

---

imagerpositiontest.set_mem_parms - Function

Package	synthesis
Module	synthesistester
Tool	imagerpositiontest

Set parameters to be used in a mem deconvolution.

Synopsis
set_mem_parms(algorithm, niter, sigma, targetflux, constrainflux, displayprogress, fixed)

Description
Set parameters to be used in a mem deconvolution. These parameters are passed to (p)imager.mem() by make_image().

Arguments

algorithm	MEM algorithm to use Allowed: String: 'entropy'|'emptiness'|'mfentropy'|'mfemptiness' Default: 'entropy'
niter	Number of mem iterations Allowed: Integer >= 0 Default: 20
sigma	Image sigma to try to achieve Allowed: Quantity (flux density) Default: '0.001Jy'
targetflux	Target flux for final image Allowed: Quantity (flux density) Default: '1.0Jy'
constrainflux	Constrain image to match target flux? else targetflux used only to initialize model Allowed: Boolean Default: F
displayprogress	Display the progress of the deconvolution Allowed: Booean Default: F
fixed	Keep model fixed? Allowed: Vector of booleans Default: F

Returns
T or fail

Example

```
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
# set up a single field mem test
- ipt.set_deconvolution_function('mem');
- ipt.set_mem_parms(algorithm='emptiness',niter=25,sigma='0.15Jy');
T
- ipt.dotests();
```

---

imagerpositiontest.set_mfcontrol_parms - Function

Package	synthesis
Module	synthesistester
Tool	imagerpositiontest

Set control parameters for multi-field imaging tests.

Synopsis
set_mfcontrol_parms(cyclefactor, cyclespeedup, stoplargenegatives, stoppointmode, scaletype, minpb, constpb)

Description
Set control parameters for multi-field imaging tests. These parameters are passed to (p)imager.setmfcontrol() by make_image when there is more than one field being imaged.

Arguments

cyclefactor	Cycle threshold = this * max sidelobe * max resid Allowed: Float Default: 3.0
cyclespeedup	Cycle threshold doubles in this number of iterations Allowed: Float Default: -1
stoplargenegatives	Stop the multiscale cycle for the first n cycles when a negative comp is found on the largest scale Allowed: Intger Default: 2
stoppointmode	Stop multiscale altogether if the smallest scale recieves this many consecutive components Allowed: Integer Default: -1
scaletype	mage plane flux scale type Allowed: String: 'SAULT'|'NONE' Default: 'NONE'
minpb	Minimum PB level to use Allowed: Float $>$ 0 Default: 0.1
constpb	In Sault weighting the flux scale is constant above this PB level Allowed: 0 $>$ Float $>$ 1 Default: 0.4

Returns

T

Example

```
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
# set up a multi field clean test using the Sault flux plane scale
- ipt.set_sim_parms(nfields=4);
- ipt.set_deconvolution_function('clean');
- ipt.set_clean_parms(algorithm='mfclark',niter=2000);
- ipt.set_mfcontrol_parms(cyclefactor=2.5,scaletype='SAULT',constpb=0.2);
T
- ipt.dotests();
```

---

imagerpositiontest.set_pimager_numprocs - Function

Package	synthesis
Module	synthesistester
Tool	imagerpositiontest

Set the number of processors available to pimager.

Synopsis
set_pimager_numprocs(numprocs)

Description
Set the number of processors available to pimager. This value gets passed to the pimager tool constructor by make_image() when the user has requested to test pimager by calling set_imaging_tool_type(pimager).

Arguments

numprocs	The number of processors available to pimager Allowed: Integer > 0 Default: 1

Returns
T

Example

```
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
# set up a single field mem test using pimager with 16 available processors
- ipt.set_imaging_tool_type('pimager');
- ipt.set_pimager_numprocs(16);
T
- ipt.set_deconvolution_function('mem');
- ipt.set_sim_parms(nfields=1,nchan=128);
- ipt.set_mem_parms(algorithm='entropy',niter=25,sigma='0.1Jy');
- ipt.dotests();
```

---

imagerpositiontest.set_seed - Function

Package	synthesis
Module	synthesistester
Tool	imagerpositiontest

Set the seed used by the random number generator for creating the simulated components.

Synopsis
set_seed(seed)

Description
Set the seed used by the random number generator for creating the simulated components. The seed is initially set to the value of time() when the tool is created. However, the user may reset it before a call to make_default_cl() is made (or before the call to dotests() which calls this function). In this way, the user can reproduce the model generated by make_default_cl(), assuming of course s/he runs on the same machine. Note that if set_sim_parms() is called with docorrupt=T, the seed for the noise generator must be set during this call for total reproducability since the simulator tool uses it's own random number generator for corrupting data.

Arguments

seed	The value to use for seeding the random number generator	
	Allowed	Float
	Default	None (is set initially during tool construction)

Returns
T

Example

```
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
# set up a single field mem that can be repeated by using the same
# random number generator seeds for the ipt tool and it's simulator tool
- ipt.set_sim_parms(docorrupt=1,seed=426);
- ipt.set_seed(-829.76);
- ipt.set_deconvolution_function('mem');
- ipt.set_mem_parms(algorithm='entropy',niter=25,sigma='0.1Jy');
- ipt.dotests();
```

---

imagerpositiontest.set_sim_parms - Function

Package	synthesis
Module	synthesistester
Tool	imagerpositiontest

Set up the parameters to be used when creating the default simulator.

Synopsis
set_sim_parms(frequency, docorrupt, scalefactor, nchan, nsources, noiselevel, nfields, deltafreq, freqresolution, sourcedirection, source_spread, minflux, maxflux, minsep, seed)

Description
Set up the parameters to be used when creating the default simulator. These parameters are used at various stages of the processing by dotests.

Arguments

frequency	The observing frequency Allowed: Quantity (frequency) Default: ’22GHz’
docorrupt	Add noise to the simulated data by calling simulator.corrupt()? Allowed: Boolean Default: F
scalefactor	The array scale factor. Controls the size of the baselines (and hence the resolution in the images) Allowed: Float > 0 Default: 1.0
nchan	Number of channels in the simulated dataset Allowed: Integer > 0 Default: 1
nsources	Number of sources to simulate Allowed: Integer > 0 Default: 4
noiselevel	The level of the noise to add when docorrupt==T Allowed: Quantity (flux density) Default: ’1Jy’
nfields	The number of fields to simulate Allowed: Integer > 0 and non prime if > 2 Default: 1
deltafreq	Channel seperation Allowed: Quantity (frequency) Default: ’1MHz’
freqresolution	Frequency resolution Allowed: Quantity (frequency) Default: ’1MHz’
sourcedirection	Direction of source Allowed: MDirection Default: dm.direction(’J2000’, ’16h00m0.0’, ’50d0m0.000’)
source_spread	Determines over how large of field the sources will be dispersed, in multiples of the field of view Allowed: Float > 0 Default: 1.0
minflux	The minimum flux density for a source. Flux densities are determined using a uniform distribution between minflux and maxflux Allowed: 0 < Float <= maxflux Default: 3.0
maxflux	The maximum flux density for a source. Flux densities are determined using a uniform distribution between minflux and maxflux Allowed: Float >= minflux Default: 3.0
minsep	The minimum source separation allowed in arcsec. No two sources will be closer than this. Note that if minsep is chosen to be too large for the field of view, source_spread, and nsources, a valid source position distribution will not be possible. In this case, make_default_cl() (or dotests()) will fail (with an informative message) Allowed: Float >= 0 Default: 5
seed	The value to use for the simulator random number generator seed Allowed: Float Default: 185349251

Returns
T or fail

Example

```
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
# set up a single field mem and corrupt the data
# set the number of sources, flux density range, and minimum
# separation as well.
- ipt.set_sim_parms(scalefactor=2,docorrupt=1,seed=426,noiselevel='0.5Jy',
+ nsources=10, minflux=20,maxflux=30,minsep=10);
- ipt.set_deconvolution_function('mem');
- ipt.set_mem_parms(algorithm='entropy',niter=25,sigma='0.1Jy');
- ipt.dotests();
```

---

imagerpositiontest.set_simulator - Function

Package	synthesis
Module	synthesistester
Tool	imagerpositiontest

Set an alternate simulator to be used for generating the MS.

Synopsis
set_simulator(sim)

Description
This function is still a bit experimental in that not everything may work is expected, such as when you set up a simulator tool to have multiple fields. Depending on the demand for this functionality, this function can be better integrated into this tool.
Set an alternate simulator to be used for generating the MS. This function allows the user to override the creation of the default simulator with his/her own simulator tool for creating the measurement set. If this function has been called successfully, dotests() will will not call make_default_simulator(). If you do not create the associated measurement set with simulator.create(), this function will do that for you using what should be reasonable defaults (ie, shadowlimit=0.001,

elevationlimit='8.0deg', autocorrwt=0.0). You can create the measurement set yourself however, in which case the tool functions will use your measurement set for processing.

Arguments

sim		The user-defined simulator to use
	Allowed	A valid simulator tool
	Default	None

Returns
T or fail

Example

```
- include 'simulator.g';
# specify a simulator
- xx := [140,180,220,330,440];
- yy := [0,-300,190,-90,60];
- zz := [0,90,-120,40,70];
- diam := 0*[1:5] + 6;
- mysim := simulator();
- mysim.settimes( integrationtime='60s', gaptime='30s', usehourangle=T,
+ starttime='0s', stoptime='7600s');
- mysim.setfield(row=1, sourcename='my_simulated_source',
+ sourcedirection=dm.direction('J2000','0h20m30','66d30m29'),
+ integrations=1);
- mysim.setconfig(telescopename='MyScope', x=xx, y=yy, z=zz,dishdiameter=diam,
+ mount='alt-az', antname='MyScope', coordsystem='local',
+ referencelocation=dm.observatory('bima'));
- mysim.setspwindow(row=1, spwname='8GHz', freq='8GHz',
+ deltafreq='1MHz', freqresolution='1MHz',nchannels=1,stokes='RR LL');
# you can choose not to create the ms yourself in which case
# set_simulator() will do it for you
- mysim.create(newms='my.ms', shadowlimit=0.01, elevationlimit='4.0deg',
+ autocorrwt=0.0);
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
- ipt.set_simulator(mysim);
- ipt.set_deconvolution_function('mem');
- ipt.set_mem_parms(algorithm='entropy',niter=25,sigma='0.1Jy');
- ipt.dotests();
```

imagerpositiontest.set_tolerance - Function

Set the tolerance allowed for a position test to pass.

Synopsis

set_tolerance(tol)

Description

Set the tolerance allowed for a position test to pass. The tolerance is specified in mupliples of beam widths. The beam widths in RA and Dec are both determined, and in order for a test to pass a found component in the restored image must be <= tolerance*beam_ra in the RA direction and <= tolerance*beam_dec in the Dec direction from the same simulated component. STILL TO COME: Ultimately, a test which involves tolerance based on signal-to-noise should be allowed, such as the canonical position error = tolerance*0.5*beamwidth/(S/N).

Arguments

tol	The allowed tolerance in beam widths AllowedFloat => 0 Default1.0

Returns

T

Example

```
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
# allow a lot of slop
- ipt.set_tolerance(5);
T
- ipt.set_deconvolution_function('mem');
- ipt.set_mem_parms(algorithm='entropy',niter=25,sigma='0.1Jy');
- ipt.dotests();
```

Compare the found source positions with the true source positions.

Synopsis

sourcecompare(complistfound, complisttrue)

Description

Compare the found and true source positions. This method is called by dotests(). It returns an array of booleans of shape [nchan,nsources], the booleans representing the result of the test.

Arguments

complistfound	The disk componentlist containing the found components Allowed: Name of disk componentlist Default: None
complisttrue	The disk componentlist containing the simulated components Allowed: Name of disk componentlist Default: None

Returns

2-d array of booleans of shape [nchan,number of components in complisttrue]

Example

```
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
# allow a lot of slop
- ipt.set_deconvolution_function('mem');
- ipt.set_mem_parms(algorithm='entropy',niter=25,sigma='0.1Jy');
```

```
- ipt.dotests();
# if you missed it the first time
- res := ipt.sourcecompare('imager_position_test/found_comp.cl',
+ 'imager_position_test/simu_comp.cl');
- res;
[[1:1,]
    T T T T]
```

---

imagerpositiontest.view - Function

Package	synthesis
Module	synthesistester
Tool	imagerpositiontest

View the restored image.

Synopsis
view()

Description
View the restored image by spawning a viewer.

Returns
T or fail

Example

```
- include 'imagerpositiontest.g';
- ipt := imagerpositiontest();
# allow a lot of slop
- ipt.set_deconvolution_function('mem');
- ipt.set_mem_parms(algorithm='entropy',niter=25,sigma='0.1Jy');
- ipt.dotests();
# you might want to actually see what you've created
- ipt.view();
T
```

1.9.3 vlaendtoend - Function

Package synthesis
Module synthesistester

Test VLA end to end processing

Synopsis
vlaendtoend()

Description
This test fills, edits, calibrates, and images a VLA mosaic of Orion at 8GHz, starting from a VLA export format file in the AIPS++ data repository. The following steps are taken:

1. The vlafiller is run on a standard VLA export file in the data repository.
2. The data are flagged using flagger to remove a known bad antenna (21) and the first ten seconds of each scan (needed for a ‘‘feature’’ of the VLA on-line system).
3. The fluxes of the calibraters are set using imager.setjy.
4. calibrater is used to calibrate the data based on the known flux calibrater and the phase calibrater.
5. flagger.filter is used to filter very anomalous points.
6. imager is used to create a mosaic of the the ten pointings in the data set.