Arguments

msname	Name of input ms Allowed:String Default:''	
imsizes	Image sizes for each round (should be monatonically increasing) Allowed:Vector of Ints Default:[128, 256]	
cellsizes	Cell sizes [in arcsec] for each round (should be monatonically decreasing) Allowed:Vector of Floats Default:[2, 1]	
scales	Name of input ms Allowed:Vector of Ints Default:[0, 5, 15]	
nitermult	Multiplier parameter used to adjust niter with imsizes Allowed:Float Default:0.15	
niterpower	Exponent of imsizes used to adjust niter with imsizes Allowed:Float Default:1.2	
fields	Vector of field ids to image Allowed:Vector of Ints Default:[1]	
spwid	Spectral window to image Allowed:Int Default:1	
centerfield	Field id for central pointing Allowed:Int Default:1	

Example

```
imagermultiscale(msname='orion.ms', imsizes=[128, 256, 512], cellsizes=[16,8,4],
scales=[0, 5, 15], nitermult=0.15, niterpower=1.2, fields=[1:4],
spwid=[1], centerfield=1);
```

1.4.15 dragon - Tool

Package	synthesis
Module	imager

Tool for wide-field synthesis imaging
include "dragon.g"

Description

dragon can perform wide-field imaging as needed to overcome the non-coplanar baselines effect for the VLA and other non-coplanar arrays. It is built upon the imager and calibrater tools for imaging and calibration. The extra capability of dragon is mainly in providing a simple and effective interface for self-calibration.

In wide-field imaging (e.g. VLA observing at 327MHz), the geometry of the array as seen from the source varies significantly over the field of view. This means that the relationship between sky brightness and visibility is no longer a simple 2D Fourier transform. However, the true relationship can be approximated by a series of tangent plane approximations in which a 2D Fourier transform is used over a small fraction of the image (usually referred to as one facet). Cleaning using a point spread function is limited to one facet at a time, and the cross-subtraction of points from one facet to another is accomplished by using a complete Fourier transform to the visibility domain.

Compared to similar programs in other packages (*i.e.* SDE dragon and the AIPS IMAGR), dragon has a number of important advantages:

- The conversion to a common tangent plane is performed in the visibility plane, and so a single output image is produced with no extra effort (this was first suggested by Sault, Staveley-Smith and Brouw, 1994).
- The decomposition of the image plane into facets may be changed during the processing, for example, starting with few facets and then proceeding to more facets at deeper levels of cleaning.
- The current clean image is displayed after each clean cycle.
- Self-calibration is performed as part of the imaging loop, thus saving time (this is also true of the SDE dragon but not of AIPS IMAGR).
- The image may be supplemented by componentmodels, which can be generated using the interactive imagerfitter. These componentmodels follow a separate path through the processing, at high precision.

- dragon can be customized by attaching commands using the AIPS++ plug-in system. See the file code/trial/apps/imager/dragon_standard.gp for an example of how to attach commands.
- Any number of outlier fields may be specified.
- dragon may be restarted and processing continued, with, for example, a different number of facets.

Example
The following example shows the quickest way to make a wide-field CLEAN image and display it. Note that this can be more easily done from the toolmanager.

```
include 'dragon.g'
#
# Wait for results before proceeding to the next step
#
dowait:=T
#
# Make an dragon tool
#
drag:=dragon('BF7.MS')
#
# Set the dragon to produce images of cellsize 30 and 2000 by 2000 pixels
#
drag.setimage(nx=2000,ny=2000, cellx='30arcsec',celly='30arcsec', nfacets=4);
#
# Weight the data
#
drag.weight(mode='briggs', pixels=2000);
#
# Make and display a clean image
#
drag.clean(niter=1000, levels='1Jy 0.3Jy 0.1Jy 0.03Jy 0.010Jy', amplitudelevel='0.015Jy',
   model='BF7.clean.model', image='BF7.clean.image')
#
dd.image('BF7.clean.image')
#
# Fourier transform the model
#
drag.ft(model='BF7.clean.model')
#
# Plot the visibilities
#
```

```
drag.plotvis()
#
# Write out the final MS and close the dragon tool
#
drag.close()
```

Constructors

dragon	Construct an dragon tool

Functions

advise	Advise (and optionally use) parameter values
done	Terminate the dragon process
filter	Apply filtering
image	Clean and self-calibrate
setimage	Set the image parameters for subsequent processing
setoutlier	Set the image parameters for an outlier field
uvrange	Apply uvrange to the visibility weights
weight	Apply weighting to the visibility weights

dragon - Constructor

Package	synthesis
Module	imager
Tool	dragon

Construct an dragon tool

Synopsis
dragon(ms)

Description
This is used to construct dragon tools associated with a MeasurementSet. The dragon tool may then be used to generate various types of images. Note that a new executable is started every time the constructor is called.
This returns a Glish variable containing the tool functions of dragon.

Arguments

ms	MeasurementSet to be processed	
	AllowedTable name	

Returns
dragon tool

Example

```
drag:=dragon('BF7.MS')
```

---

dragon.setimage - Function

Package	synthesis
Module	imager
Tool	dragon

Set the image parameters for subsequent processing

Synopsis
setimage(name, nx, ny, cellx, celly, stokes, doshift, phasecenter, shiftx, shifty, mode, nchan, start, step, spwid, fieldid, facets, name)

Description
Define the default image parameters.
Note that some parameters can be specified either in canonical units or via measures. To establish default values, the ids for the default spectral window and default field id must be given. The meaning of arguments mode, nchan, step, *etc.* is described in setdata. dragon can perform multi-frequency synthesis over several spectral windows (mode='mfs'). To acheive this, you should set spwid to an array of the required spectral windows (*e.g.* spwid=1:2).
The phase center of the image defaults to that of the specified fieldid (the first if none is specified). Note that fieldid refers to the field identifier in the MeasurementSet and is not connected to the number of facets in any way. A phase center may be also specified in an argument to setimage using any valid direction. If the conversion from the observed direction requires frame information then this is taken as follows:

- Direction information, including the coordinate system, is taken from the relevant entry in the Field table of the MeasurementSet.

- The epoch is taken from the time of observation of each visibility.

- A position is specified via the dragon tool function setoptions

If the specified number of facets is greater than unity then the image is split into facets (this number along the x and y axes) and processed.

Finally, a position shift may be added using the arguments shiftx, shifty. This will be added to whatever the phase center was set to as described above. The shift is a real angle on the dragon so that, in e.g. RA, DEC, the RA shift is divided by cos(DEC) before being added to the RA. The sense of the shift is that the position after addition of the shift gives the new phase center of the image. The shift is in the reference frame of the phase center.

Arguments

name	Name of output image Allowed: String Default: dragon
nx	Number of spatial pixels in x Allowed: Int Default: 128
ny	Number of spatial pixels in y Allowed: Int Default: 128
cellx	Cellsize in x (e.g. '1arcsec') Allowed: Quantity Default: '1arcsec'
celly	Cellsize in y (e.g. '1arcsec') Allowed: Quantity Default: '1arcsec'
stokes	Stokes parameters to image (e.g. 'IQUV') Allowed: 'I'|'IV'|'IQU'|'IQUV' Default: 'I'
doshift	Use the specified phase center? T or F Allowed: Bool Default: T
phasecenter	Direction of phase center as a measure Allowed: MDirection
shiftx	Shift in x (e.g. '23.7arcsec') Allowed: Quantity Default: '0arcsec'
shifty	Shift in y (e.g. '-54.2arcsec') Allowed: Quantity Default: '0arcsec'
mode	Type of processing Allowed: 'mfs'|'channel'|'velocity' Default: 'mfs'
nchan	Number of channels Allowed: Int Default: 1
start	Start channel (1-relative) Allowed: Int
step	Step in channel Allowed: Int
spwid	Spectral Window Ids (1 relative) Allowed: Vector of Ints Default: 1
fieldid	Field Id (1 relative) Allowed: Int Default: 1
facets	Number of facets Allowed: Integer Default: 1
name	Name of output image Allowed: String Default: dragon

---

dragon.setoutlier - Function

Package	synthesis
Module	imager
Tool	dragon

Set the image parameters for an outlier field

Synopsis
setoutlier(name, nx, ny, cellx, celly, stokes, doshift, phasecenter, mode, nchan, start, step, spwid, fieldid)

Description
Define the image parameters for an outlier field. Any number of outlier fields may be defined, but each must have a unique name.

The phase center of the image defaults to that of the specified fieldid (the first if none is specified). A phase center may be also specified in an argument to setimage using any valid direction. If the conversion from the observed direction requires frame information then this is taken as follows:

- Direction information, including the coordinate system, is taken from the relevant entry in the Field table of the MeasurementSet.
- The epoch is taken from the time of observation of each visibility.
- A position is specified via the dragon tool function setoptions

Arguments

name	Name of output image Allowed String Default dragon.outlier.n
nx	Number of spatial pixels in x Allowed Int Default 128
ny	Number of spatial pixels in y Allowed Int Default 128
cellx	Cellsize in x (e.g. '1arcsec') Allowed Quantity Default '1arcsec'
celly	Cellsize in y (e.g. '1arcsec') Allowed Quantity Default '1arcsec'
stokes	Stokes parameters to image (e.g. 'IQUV') Allowed 'I'|'IV'|'IQU'|'IQUV' Default 'I'
doshift	Use the specified phase center? T or F Allowed Bool Default T
phasecenter	Direction of phase center as a measure Allowed MDirection
mode	Type of processing Allowed 'mfs'|'channel'|'velocity' Default 'mfs'
nchan	Number of channels Allowed Int Default 1
start	Start channel (1-relative) Allowed Int
step	Step in channel Allowed Int
spwid	Spectral Window Ids (1 relative) Allowed Vector of Ints Default 1
fieldid	Field Id (1 relative) Allowed Int Default 1

---

dragon.advise - Function

Package synthesis
Module imager
Tool dragon

Advise (and optionally use) parameter values

Synopsis
advise(amplitudeloss, fieldofview, cell, facets, phasecenter)

Description
Advise on recommended values of certain parameters.
The calculations are performed as following:

cell The maximum uv distance in wavelength is found and then half of the inverse is taken as the maximum cellsize allowed.

pixels The field of view is converted to a number of pixels using the calculated cell size.

facets The number of facets on an axis is calculated in two different ways. The first method simply requires that the peeling of facets away from the celestial sphere should not cause an amplitude drop of more than the argument amplitudeloss. The positions may be incorrect, but all the sources will be removed correctly. The second method requires that the source positions be accurate to the same fraction of the beam specified by amplitudeloss. The second calculates the second moment in w and in uv distance and chooses the number of facets correspondingly. The first method does the same but after fitting a plane to the sampling: $w = au + bv$. For an approximately coplanar array, the positions may be wrong but the removal of sidelobes will be accurate. The number of facets returned is the second, usually smaller, number. The formula used is:

$$N_{facets} = N_{pixels} \sqrt{\frac{\frac{\Delta\theta}{\sqrt{8\delta A}} w_{rms}}{uv_{rms}}} \tag{1.11}$$

where $\Delta\theta$ is the cellsize in radians, and δA is the amplitude loss. This formula can be derived from (a) the peeling of facets from the celestial sphere, and (b) a quadratic approximation for the beam size both in the plane of the sky and along the w axis.

Arguments

amplitudeloss	Maximum fractional amplitude loss due to faceting Allowed: Float Default: 0.05
fieldofview	Desired field of view Allowed: Quantity Default: '1deg'
cell	Recommended maximum cellsize Allowed: Quantity
facets	Recommended number of facets on one axis Allowed: Integer Default: 1
phasecenter	Direction of phase center as a measure Allowed: MDirection

---

dragon.weight - Function

Package	synthesis
Module	imager
Tool	dragon

Apply weighting to the visibility weights

Synopsis
weight(type, rmode, noise, robust, fieldofview, npixels, async)

Description
Apply visibility weighting to correct for the local density of sampling in the uv plane. The imaging weights are written into a Table column called IMAGING_WEIGHT, which may be plotted using plotweights. In addition this columns may be accessed directly using either the table or ms modules.
To correct for visibility sampling effects, natural, uniform (the default), radial, and briggs weighting are supported. These work as follows. Then:

natural : minimizes the noise in the dirty image. The weight of the i-th sample is set to the inverse variance:

$$w_i = \frac{1}{\sigma_i^2} \tag{1.12}$$

where σ_i is the noise of the i'th sample.

radial : approximately minimizes rms sidelobes for an east-west synthesis array. The weight of the i-th sample is multiplied by the radial distance from the center of the u,v plane:

$$w_i = w_i \sqrt{u_i^2 + v_i^2} \tag{1.13}$$

uniform : For briggs and uniform weighting, we first grid the inverse variance w_i for all selected data onto a grid of size given by the argument npixels (default to nx) and u,v cell-size given by 2/fieldofview where fieldofview is the specified field of view (defaults to the image field of view). This forms the gridded weights W_k. The weight of the i-th sample is then changed:

$$w_i = \frac{w_i}{W_k} \tag{1.14}$$

where W_k is the gridded weight of the relevant cell. It may be shown that this minimizes rms sidelobes over the field of view. By changing the field of view, one may suppress the sidelobes over a region different (usually smaller) than the image size.

briggs: rmode='norm' : The weights are changed:

$$w_i = \frac{w_i}{1 + W_k f^2} \tag{1.15}$$

where:

$$f^2 = \frac{(5 * 10^{-R})^2}{\frac{\sum_k W_k^2}{\sum_i w_i}} \tag{1.16}$$

and R is the briggs parameter. The scaling of R is such that $R = 0$ gives a good tradeoff between resolution and sensitivity.

briggs: rmode='abs' : The weights are changed:

$$w_i = \frac{w_i}{W_k * R^2 + 2 * \sigma_R^2} \tag{1.17}$$

where R is the briggs parameter and σ_R is the noise parameter.

For more details about briggs weighting, see the Briggs thesis. Note that this weighting is *not* cumulative since the imaging weights are calculated from the specified sigma (expected noise) per visibility (actually stored in the SIGMA column).

Arguments

type	Type of weighting Allowed String: 'uniform' | 'natural' | 'briggs' | 'radial' Default uniform
rmode	Mode of briggs weighting Allowed String: 'norm' | 'abs' | 'none' Default none
noise	Noise used in absolute briggs weighting Allowed Quantity Default '0.0Jy'
robust	Parameter in briggs weighting Allowed Double Default 0.0
fieldofview	Field of view for uniform weighting Allowed Quantity Default '0arcsec'
npixels	Number of pixels in the u and v directions Allowed Int Default 0
async	Run asynchronously in the background Allowed Bool Default !dowait

Returns
Bool

Example

```
drag.weight(type='briggs', rmode='norm', robust=0.5)
```

Applies briggs weighting.

---

dragon.filter - Function

Package	synthesis
Module	imager
Tool	dragon

Apply filtering

Synopsis
filter(bmaj, bmin, bpa, async)

Description
Apply visibility tapering to emphasize certain scale structures. The imaging tapers are applied to a Table column called IMAGING_WEIGHT, which may be plotted using plotweights. In addition, this column may be accessed directly using either the table or ms modules. Note that the taper is multiplicative and so the weights must be calculated first using weight. The points are not flagged!
Note that the scale size to be emphasized is given in the image plane as the parameters of the correspondiong Gaussian.

Arguments

bmaj	Major axis of filter Allowed: Quantity Default: '0rad'
bmin	Minor axis of filter Allowed: Quantiy Default: '0rad'
bpa	Position angle of filter Allowed: Quantity Default: '0deg'
async	Run asynchronously in the background Allowed: Bool Default: !dowait

Returns
Bool

Example

```
drag.weight('uniform')
drag.filter(type='gaussian', bmaj='2.3arcsec', bmin='1.67arcsec',
bpa='-34.5deg')
```

---

dragon.uvrange - Function

Package	synthesis
Module	imager
Tool	dragon

Apply uvrange to the visibility weights

Synopsis
uvrange(uvmin, uvmax)

Description
Apply a uvrange so that only points within a given uvrange have non-zero weight in the IMAGING_WEIGHT column, which may be plotted using plotweights. The points are not flagged!

Arguments

uvmin	Minimum uv distance allowed (wavelengths) Allowed Float Default 0.0
uvmax	Maximum uv distance allowed (wavelengths) Allowed Float Default 0.0

Returns
Bool

Example

```
drag.weight('uniform')
drag.uvrange(0, 4000.0)
```

---

dragon.image - Function

Package	synthesis
Module	imager
Tool	dragon

Clean and self-calibrate

Synopsis
image(levels, amplitudelevels, timescales, niter, gain,

threshold, model, complist, image, residual, statsregion, statsout, algorithm, maskmodification)

Description

Makes a clean, self-calibrated image using the wide-field algorithms.

For **cleaning**, the deconvolution is performed on the residual image calculated from the visibility data currently selected. Hence the first step performed in clean is to transform the current model or models (optionally including a componentlist) to fill in the MODEL_DATA column, and then inverse transform the residual visibilities to get residual image for all facets. These residual image are then cleaned using the corresponding point spread function. This means that if the initial model is used as the starting point for the deconvolution. Thus if you want to restart a clean, simply set the model to the model that was previously produced by clean.

The CLEAN deconvolution is joint in whatever Stokes parameters are present. Thus it searchs for peaks in either I or $I + |V|$ or $I + \sqrt{Q^2 + U^2 + V^2}$, the rationale for the latter two forms being to be biased towards finding strongly polarized pixels first (these forms are also the maximum eigenvalue of the coherency matrix). The PSF is constrained to be the same in all polarizations (a feature of this implementation, not of the Hamaker-Bregman-Sault formalism).

The clean algorithm is split into minor and major cycles. In the minor cycles only the brightest points are cleaned, using a subset of the point spread function. In the major cycle, the points thus found are subtracted from the original visibilities. Note that aliasing can be reduced by using the padding argument in setoptions.

In the **self-calibration** phase, the data are self-calibrated to reduce the rms difference between the current model and the observed data. The antenna gain model may be either phase-only via the *T-Jones* or amplitude and phase via *G-Jones* (see calibrater for more details).

Self-calibration is invoked whenever the peak residual (in any field) drops below the current level (as specified in the levels argument). The best strategy is usually to set levels to initiate the first self-calibration when close to but above the worst calibration errors in the image, and thereafter down in steps of a factor of about 3. Usually at the high flux levels, only phase self-calibration is required. One can switch to amplitude and phase self-calibration by setting the argument amplitudelevel appropriately.

Note that the processing can be restarted from the existing images. Note also that the number of facets may be changed on restart.

There are 2 options of masking, for defining regions in which to look for CLEAN components, available. If the maskmodification argument is set to 'auto', when the image is cleaned to the flux level of a calibration stage, an automatic mask is made by thresholding the restored image to the flux level reached. However if maskmodification is set to 'interactive' a viewer tool is started prompting the user to draw the mask (See interactivemask).

Arguments

levels	Flux levels at which to self-calibrate e.g. 0.3Jy 0.1Jy 0.03Jy Allowed:Vector of strings Default:'0Jy'
amplitudelevels	Flux level below which amplitude self-calibration is used Allowed:Quantity Default:'0Jy'
timescales	Time scales for the self-calibrations e.g (60s 10s 10s) Allowed:Vector of strings Default:'10s'
niter	Number of Iterations, set to zero for no CLEANing Allowed:Int Default:1000
gain	Loop Gain for CLEANing Allowed:Double Default:0.1
threshold	Flux level below which cleaning will stop Allowed:Quantity Default:'0Jy'
model	Name of images Allowed:Vector of strings
complist	Name of component list Allowed:String
image	Names of restored images Allowed:Vector of strings
residual	Names of restored images Allowed:Vector of strings
statsregion	Region in which to calculate statistics Allowed:Any valid region Default:unset
statsout	Statistics record Allowed:Record Default:[=]
algorithm	deconvolving algorithm Allowed:String Default:'wfclark'
maskmodification	Masking update at every cal cycle from 'none', 'auto', 'interactive' Allowed:String Default:'none'

Returns

Bool

Example

```
drag.image(levels='0.3Jy 0.1Jy 0.03Jy 0.01Jy', amplitudelevel='0.02Jy', model='BF7.clean.model',
image='BF7.clean.restored', niter=1000, gain=0.25)
```

---

dragon.done - Function

Package	synthesis
Module	imager
Tool	dragon

Terminate the dragon process

Synopsis
done()

Description
This is used to totally stop the dragon process. It is a good idea to conserve memory use on your machine by stopping the process once you no longer need it.

Returns
Bool

1.4.16 dragontest - Function

Package	synthesis
Module	imager

Test dragon using a VLA 74 MHz data set

Synopsis
dragontest(fitsfile, doimage, dofind)

Description
This runs dragon on a simulated data set formed from a standard VLA 74MHz data set called '3DDAT'. This data set can be obtained from Tim Cornwell.
The steps are:

1. The FITS file is converted to a MeasurementSet
2. A componentlist of seven components is constructed
3. Simulated data for the components with phase errors of 0.5 degree and 1% amplitude errors.
4. An image is constructed using dragon
5. Point sources are found and compared to the input componentlist

Arguments

fitsfile	Name of standard FITS file Allowed String Default 3DDAT
doimage	Do the imaging as well as the simulation? Allowed Boolean Default T
dofind	Do the source finding as well as the imaging and simulation? Allowed Boolean Default T

Returns
Bool

1.4.17 imagerwizard - Function

Package synthesis
Module imager

Wizard for synthesis imaging

Synopsis
imagerwizard()

```
include "imagerwizard.g"
```

Description

imagerwizard provides a GUI for making a clean image from a MeasurementSet with the minimum of user intervention. It is based on the table, imager, ms, and wizard Display is performed using the viewer.
In imagerwizard, the user is walked through a number of steps using a simple GUI:

- Either a MeasurementSet or a FITS file can be specified. If the latter, it is converted to a MeasurementSet using the fitstoms constructor of the ms tool.
- The spectral windows are listed and the user asked to select one for imaging.
- The user is asked to select values for certain control values. Note that all of these can be left as the default values (*i.e.* mostly unset values).
- A clean image of the entire primary beam is made from the inner 33% of the uv plane using 1000 Clean iterations.
- The resulting clean image is displayed and the user asked to zoom the region of interest.
- The Clean is repeated at full resolution with all the data, cleaned down to 3 σ where the noise is estimated as three times the Stokes V rms.
- The final image is displayed.

Typically, imagerwizard works well for a quick look at all synthesis data except VLBI but you may want to use imager directly for the final image.
During the cleaning stage, some of the relevant Glish commands for imager and the other tools are displayed. Note that these are slightly more verbose than you would type but should be executable. Not all commands are displayed, obviously. In particular, we have not shown those related to the operation of imagerwizard. Hence all the command you see are ones that you might plausibly use during data reduction. The commands are written to a file called 'imagerwizard_commands.g' in the current directory.
Note that if you don't have a dataset, then the function imagermaketestms() can be used to make a copy of the standard

test MeasurementSet 3C273XC1.ms. The button *Make test MS* (in stage 1) will accomplish the same thing.

Example

```
include 'imagerwizard.g'
imagermaketestms() # To make the standard MS 3C273XC1.ms
imagerwizard() # To start the GUI
```

1.4.18 mosaicwizard - Function

Package synthesis
Module imager

Wizard for multi-field synthesis imaging

Synopsis
mosaicwizard()
include "mosaicwizard.g"

Description

mosaicwizard provides a GUI for making a deconvolved image from a multi-field Measurement Set or FITS file with the minimum of user intervention. It is based on table, imager, ms, and wizard. Display is performed using the viewer.
In mosaicwizard, the user is walked through a number of steps using a simple GUI to create first a low resolution image, and then higher resolution images until the full resolution is achieved. At each step of increasing resolution, the previous low resolution image is regridded and used as a starting model, so only incremental changes from this model are solved for. This technique is not very effective for dealing with point sources, but it is excellent at imaging extended structure. The steps which the mosaicwizard lead you through include:

- Either a MeasurementSet or a FITS file can be specified. If the latter, it is converted to a MeasurementSet using the fitstoms constructor of the ms tool.
- The user can provide an optional initial model.

- The spectral windows are listed and the user asked to select one for imaging.
- The positions of the fields are plotted, and the user is asked to select several for imaging. The extent of the image mosaicwizard makes is initially set by the fields selected and the calculated extent of the primary beam.
- The main imaging loop is described below:
  - The user is asked to select the deconvolution algorithm and the values for certain control parameters. The most important of these is the (u, v) scaling parameter (which controls the maximum baseline used as a fraction of the maximum baseline present in the data, hence determining the resolution of the mosaic image). Some deconvolution algorithms, such as MEM and Multi-Scale Clean, require additional specialized parameters. Note that the image size and cell size are not directly under user control. However, they are indirectly controlled by the region size (see below) and the (u, v) scaling parameter.
  - A low resolution deconvolved mosaic of all selected fields is made. In this stage, the data are weighted as determined by the weighting parameter, plus an additional Gaussian taper if the image is not at full resolution. The Gaussian taper will reduce PSF sidelobes caused by the sharp cutoff in the Fourier plane coverage when the (u, v) scaling parameter is less than 1. If you set the displayprogreess parameter to be true, you will get a real time display showing the residuals and the cleaned flux as a function of iteration number. This progress display can be very helpful in understanding what is happening in the deconvolution, and in analizing any failures which may occur.
  - The resulting image is displayed and the user asked to zoom the region of interest. Either a rectangular or a polygonal region can be selected. The region will have two effects on the imaging: first, the angular size of the next iteration's image will be smaller, matched to this region (plus an appropriate guard band as required by the PSF; note that mosaics with many pointings don't need the traditional factor of 2 PSF guard band); second, the region of interest will also be turned into a mask image which will be used like a CLEAN box in the next iteration's deconvolution, precluding deconvolved

emission from the region outside the mask. Note that the details of the mask generation are not included in the generated script, as they require interactive user input. Making a good tight mask can be very important in some mosaics, especially if you lack total power. For example, in order to get a good image out of the test Cas A MeasurementSet, you need to supply a good mask early on in the process.

  - The imaging loop is automatically repeated, with the (u, v) scaling parameter increasing by a factor of 2 by default, though the user may wish to change this. The imaging loop stops when the scaling parameter is greater or equal to 1, or when full resolution has been reached. The model image generated from the iteration of the loop we just completed will be regridded to the next iteration's resolution, clipped at zero, and used as the initial model for the next iteration.

Typically, mosaicwizard works well for a quick look at the mosaic data, but you may want to use imager directly for the final image if you find that mosaicwizard does not give you enough control over the imaging process.
During the deconvolution stage, many of the relevant Glish commands for imager and the other tools are displayed. Note that these are slightly more verbose than you would type but should be executable. Not all commands are displayed, obviously. In particular, we have not shown those related to the operation of mosaicwizard or those related to the generation of the mask (which requires interactive user input). Hence all the commands you see are ones that you might plausibly use during data reduction. The commands are written to a file called 'scripter.log_*' in the current directory. A more complete listing of the commands will be forthcoming in the next release.
Note that if you don't have a data set, then the function imagermaketestmfms() can be used to make a copy of the standard mosaicing test MeasurementSet XCAS.ms, a seven pointing VLA X band D array data set on Cas A. If you leave the MeasurementSet parameter unset in step 1 of the mosaicwizard, the mosaicwizard will create a MeasurementSet called mosaicwizard.ms from the Cas A data.

Example

```
include 'mosaicwizard.g'
mosaicwizard() # To start the GUI
```