Package synthesis
Module simulator
Tool imageevaluator

Finishes up the imageevaluator tool

Synopsis
done()

Description
Done kills off the image and frees up memory.

Returns
Bool

Example

```
ime.done();
```

---

include "dynamicsched.g"

1.7.3 dynamicsched - Tool

Package synthesis
Module simulator

tool for simulating dynamic scheduling

Description

dynamicsched provides a set of simple tools for exploring the efficiency of dynamic and static scheduling. It requires site testing data including phase monitor and opacity data, as it available for many sites now (such as the Chajnantor site in Chile).

Example
The following example shows how to use the dynamicsched dynamic
scheduling simulator.

```
include 'dynamicsched.g'
mysched := dynamicsched();
#
# info about teh site testing data
#
mysched.setsitedata(sitedatafile='CH.9506.INT+ALL', headerfile='header.sitedata',
observatory='ALMA', seeinglambda='26.79mm', seeingel='36deg', seeingbaseline='300m'));
#
# theoretical opacity profile
#
mysched.settaudata(tautermsfile='CH.LIEBE.TERMS');
#
# perform scheduling for 100 hours, starting 1 day after the start of the site data file;
# schedule increment is 0.5 hours.
#
mysched.settimes(dt='0.5h', relativestart='1.00d', duration='100.0h')
mysched.setschedcriteria();
mysched.setphasecalinfo();
mysched.setchangeinfo(  azslewrate='2.0deg/s', elslewrate='1.0deg/s',
changeoverhead='60s');
#
# define the sensitivity of the instrument
#
mysched.setsensitivity(dishdiameter='12m', nantennas=64, npol=2);
mysched.setbandsensitivity(1, 'band1', '1GHz', '100GHz', '8GHz', '30K', 0.80);
mysched.setbandsensitivity(2, 'band2', '100GHz', '300GHz', '8GHz',  '50K', 0.75);
mysched.setbandsensitivity(3, 'band3', '300GHz', '500GHz',  '8GHz', '100K',  0.70);
mysched.setbandsensitivity(4, 'band4', '500GHz', '700GHz',  '8GHz', '200K', 0.60);
mysched.setbandsensitivity(5, 'band5', '700GHz', '1000GHz',  '8GHz', '300K',  0.50);
mysched.viewsensitivity();
#
# create many simulated projects; can control distribution of observing time,
# frequency, etc
#
mysched.generateprojects(nprojects=100, timemode='2h', timesigma='5h', timemax='20h');
mysched.probview('rating')
mysched.probview('freq')
mysched.probview('radec')
mysched.probview('time')
mysched.saveprojects(projecttable='PROJECTS.SAVE');
#
# actually make a schedule for our simulated projects
```

```
#
mysched.schedule();
mysched.saveschedule(scheduletable='SCHEDULE.SAVE');
mysched.done();
#
```

Constructors

dynamicsched	Construct an dynamic scheduling

Functions

defaultinitialize	Perform a default initialization -- archaic
done	Clean up the dynamic scheduling object
evaluateobservations	Evaluate the observations
generateprojects	Create many simulated observing projects
getpgplotter	Get the pgplotter
probview	View the probability distributions for time, freq, rating, RA/DEC
recoverprojects	Recover the observing projects from a Table
recoverschedule	Recover an observing schedule from a table
reobserveschedule	Given a schedule reobserve it for the current conditions
saveprojects	Save the observing projects to a Table
saveschedule	Save the current observing schedule to a table
schedule	Go through the site test data and schedule projects
setbandsensitivity	Specify band-dependent parameters which influence the sensitivity
setchangeinfo	Specify parameters which affect how quickly source changes occur
setphasecalinfo	Provide parameters which determine how well the phase calibration works
setschedcriteria	Set criteria for scheduling observations
setsensitivity	Specifiy coarse telescope parameters which influence sensitivity
setsitedata	Define a few site related data
settaudata	Name the file with the model of opacity vs frequency
settimes	Specify time-related parameters
viewsensitivity	Look at the sensitivity parameters that have been set

dynamicsched - Constructor

Package	synthesis
Module	simulator
Tool	dynamicsched

Construct an dynamic scheduling

Synopsis
dynamicsched()

Description
This is used to construct a dynamicsched tool. This returns a Glish variable containing the tool functions of dynamicsched.

Arguments

Returns
dynamicsched tool

Example

```
mysched:=dynamicsched();
```

---

dynamicsched.setsitedata - Function

Package	synthesis
Module	simulator
Tool	dynamicsched

Define a few site related data

Synopsis
setsitedata(sitedatafile, headerfile, observatory, seeinglambda, seeingel, seeingbaseline)

Description
Set the site position and the file containing the site survey data in ALMA standard ASCII format. Also set the details of the seeing (ie, phase) monitor.

Arguments

sitedatafile	ASCII site survey file Allowed:String Default:'CH.9506.INT+ALL'
headerfile	Header file associated with sitedatafile Allowed:String Default:'header.sitedata'
observatory	Name of an existing site Allowed:String Default:'ALMA'
seeinglambda	Observing wavelength of seeing monitor Allowed:Quantity Default:'26.79mm'
seeingel	Elevation angle of seeing monitor's beam Allowed:Quantity Default:'36deg'
seeingbaseline	Baseline length of seeing monitor Allowed:Quantity Default:'300m'

Returns
Bool

Example

```
mysched.setsitedata(sitedatafile='CH.9506.INT+ALL',
headerfile='header.sitedata', observatory='ALMA', seeinglambda='3cm',
seeingel='30deg', seeingbaseline='300m');
```

---

dynamicsched.settaudata - Function

Package	synthesis
Module	simulator
Tool	dynamicsched

Name the file with the model of opacity vs frequency

Synopsis
settaudata(tautermsfile)

Description

This ASCII file, with built in header, contains columns for frequency (in GHz), and the dry, and the wet terms. Currently, the dynamicsched tool expects the frequencies to range from 1 to 1000 GHz in increments of 1 GHz.

Arguments

tautermsfile	File containing tau data Allowed String Default 'CH.LIEBE.TERMS'

Returns
Bool

Example

```
mysched.settauterms(tautermsfile='CH.LIEBE.TERMS');
```

---

dynamicsched.setchangeinfo - Function

Package	synthesis
Module	simulator
Tool	dynamicsched

Specify parameters which affect how quickly source changes occur

Synopsis
setchangeinfo(azslewrate, elslewrate, changeoverhead)

Description
Specify parameters which affect how quickly source changes can occur.

Arguments

azslewrate	Slew rate in azimuth axis Allowed:Quantity Default:'2deg/s'
elslewrate	Slew rate in elevation axis Allowed:Quantity Default:'1deg/s'
changeoverhead	Time lost to setup and calibration when switching projects Allowed:Quantity Default:'60s'

Returns
Bool

Example

```
mysched.setchangeinfo(azslewrate='2deg/s', elslewrate='2deg/s',
changeoverhead='60s')
```

---

dynamicsched.settimes - Function

Package	synthesis
Module	simulator
Tool	dynamicsched

Specify time-related parameters

Synopsis
settimes(dt, timeref, absolutestart, relativestart, duration)

Description
We specify the granularity of the dynamic scheduling (which should not be finer than the time between each site survey data point), and the start and stop times for which the user wants to make a dynamic schedule.

Arguments

dt	Granularity of scheduling Allowed:Quantity Default:'0.25h'
timeref	relative or absolute starting time Allowed:'relative'|'absolute' Default:'relative'
absolutestart	Starting time as an absolute epoch Allowed:Epoch Default:dm.epoch('utc', 'today')
relativestart	Starting time relative to first site data point Allowed:Quantity Default:'0.0d'
duration	Duration of scheduling (<0 means till sitedata runs out) Allowed:Quantity Default:'1.0d'

Returns
Bool

Example

```
mysched.settimes(dt='600s', timeref='relative', relativestart='0.0d',
duration='-1');
```

---

dynamicsched.setphasecalinfo - Function

Package	synthesis
Module	simulator
Tool	dynamicsched

Provide parameters which determine how well the phase calibration works

Synopsis
setphasecalinfo(phasecalmethod, baselevel, fraclevel, windvelocity, timescale)

Description

The residual phase errors after calibration by the relevant method are a topic of debate, especially for radiometric calibration. For radiometric phase calibration, we assume the residual phase errors σ_ϕ are given by

$$\sigma_\phi = \sqrt{b^2 + f^2 \cdot D_\phi(vt)}, \tag{1.35}$$

where b is some base error level related to the radiometer sensitivity, f is the fraction of atmospheric phase errors which are not corrected, D_ϕ is the phase structure function, v is the wind velocity, and t is a time scale for the atmospheric phase errors.

Arguments

phasecalmethod	Phasecal method to use (only RADIOMETRIC works at the moment Allowed: 'RADIOMETRIC' | 'FASTSWITCH' Default: 'RADIOMETRIC'
baselevel	Basic accuracy of the phase correction (in pathlength) Allowed: Quantity Default: '50um'
fraclevel	Fraction of the atmospheric phase errors which are not removed Allowed: Float Default: 0.10
windvelocity	Velocity of winds aloft Allowed: Quantity Default: '10m/s'
timescale	Timescale of the atmospheric phase errors Allowed: Quantity Default: '30s'

Returns
Bool

Example

```
mysched.setphasecalinfo(phasecalmethod='RADIOMETRIC',
baselevel='50um', fraclevel=0.1,  timescale='20s')
```

---

dynamicsched.setsensitivity - Function

Package synthesis
Module simulator
Tool dynamicsched

Specifiy coarse telescope parameters which influence sensitivity

Synopsis
setsensitivity(dishdiameter, nantennas, npol)

Description
Specifiy coarse parameters which influence the array sensitivity, ie, the number of antennas, their size, and the number of polarizations.

Arguments

dishdiameter	Size of antennas Allowed: Quantity Default: '12m'
nantennas	Number of antennas Allowed: Int Default: 64
npol	Number of polarizations Allowed: Int Default: 2

Returns
Bool

Example

```
mysched.setsensitivity(dishdiameter='12m', nantennas=64, npol=2);
```

---

dynamicsched.setbandsensitivity - Function

Package synthesis
Module simulator
Tool dynamicsched

Specify band-dependent parameters which influence the sensitivity

Synopsis

```
setbandsensitivity(whichband, bandname, freqlow, freqhigh,
bandwidth, tsys, efficiency)
```

Description

Specify band-dependent parameters which influence the sensitivity, such as antenna efficiency, bandwidth, and system temperature. Opacity and decorrelation are handled automatically through the site testing data.

Arguments

whichband	Give the band number (1 relative) Allowed: Int Default: 1
bandname	Band name (just used as a label Allowed: String Default: 'band1'
freqlow	Low frequency band limit Allowed: Quantity Default: '100GHz'
freqhigh	High frequency band limit Allowed: Quantity Default: '1000GHz'
bandwidth	Bandwith for the observations Allowed: Quantity Default: '8GHz'
tsys	System temperature (below the atmosphere) Allowed: Quantity Default: '100K'
efficiency	Cumulative efficiency (antenna and correlator) Allowed: Float Default: 0.80

Returns

Bool

Example

```
mysched.setbandsensitivity(1, 'band1', '1GHz', '100GHz', '8GHz', '30K',
0.80);
mysched.setbandsensitivity(2, 'band2', '100GHz', '300GHz', '8GHz',  '50K',
0.75);
mysched.setbandsensitivity(3, 'band3', '300GHz', '500GHz',  '8GHz', '100K',
0.70);
```

---

dynamicsched.viewsensitivity - Function

Package	synthesis
Module	simulator
Tool	dynamicsched

Look at the sensitivity parameters that have been set

Synopsis
viewsensitivity()

Description
As you set the band sensitivity, you may forget what you've done. You can always take a look at the status of the sensitivity model at hand by invoking viewsensitivity.

Arguments

Returns
Bool

Example

```
mysched.viewsensitivity();
```

---

dynamicsched.generateprojects - Function

Package	synthesis
Module	simulator
Tool	dynamicsched

Create many simulated observing projects

Synopsis

generateprojects(nprojects, ratingmin, ratingmax, timemode, timesigma, timemax, freqtransexponent, freqwt, freqexponent, decmin, decmax)

Description

This function generates simulated observing projects which a particular frequency distribution, observing time distribution, rating distribution, and RA/DEC distribution.

Currently, the model for the observing frequency distribution is

$$trans^{transexp} * (1 + freqwt * (freq/freqmax)^{freqexp}), \tag{1.36}$$

where $trans$ is the atmospheric transmission at 1 mm of precipitable water vapor. The first term provides for the basic trend that you observe more often where the atmosphere is more transparent. The freqwt and freqexp permit the simulator to place more weight on higher frequencies, where the atmosphere is less transparent but the science may be deemed to be more interesting.

The observing time distribution is a Gaussian with peak located at timemode, a width of timesigma, a minimum observing time of one hour, a maximum observing time of timemax, and quantized in integer numbers of hours.

The project rating distribution is flat between ratingmin and ratingmax, with only integer values.

The distribution in RA/DEC is assumed to be flat (ie, fewer sources at hogh and low declination), more at the equator. The declination range is set by decmin and decmax.

The distributions of ratings, observing frequencies, observing times, and RA/DEC are assumed to be independent.

Arguments

nprojects	Number of projects to generate Allowed:Int Default:100
ratingmin	Minimum rating Allowed:Int Default:1
ratingmax	Maximum rating Allowed:Int Default:10
timemode	Peak of distribution of allotted observing times Allowed:Quantity Default:'5h'
timesigma	Width of distribution of allotted observing times Allowed:Quantity Default:'10h'
timemax	Extent of tail of distribution of allotted observing times Allowed:Quantity Default:'20h'
freqtransexponent	Exponent of atmospheric transmission in freq distribution Allowed:Quantity Default:2.0
freqwt	Multiplier to increase weight of high frequency observations Allowed:Quantity Default:2.0
freqexponent	Exponent to increase weight of high frequency observations Allowed:Quantity Default:1.5
decmin	Minimum declination in dec range Allowed:Quantity Default:'-90deg'
decmax	Maximum declination in dec range Allowed:Quantity Default:'52deg'

Returns
Bool

Example

```
mysched.generateprojects(nprojects=300, timemean='1h',
timesigma='10h', timemax='30h', freqwt=1.0);
```

---

dynamicsched.probview - Function

Package	synthesis
Module	simulator
Tool	dynamicsched

View the probability distributions for time, freq, rating, RA/DEC

Synopsis
probview(whichone)

Description
View the probability distributions for time, freq, rating, RA/DEC which have been created by generateprojects and are reflected in the projects.

Arguments

whichone	Which distribution should we look at? Allowed:'freq'|'rating'|'radec'|'time' Default:'freq'

Returns
Bool

Example

```
mysched.probview('freq');
mysched.probview('time');
mysched.probview('radec');
mysched.probview('rating');
```

---

dynamicsched.saveprojects - Function

Package	synthesis
Module	simulator
Tool	dynamicsched

Save the observing projects to a Table

Synopsis
saveprojects(projecttable, allprojects, whichprojects)

Description
Save the observing projects to a Table. Information such as the amount of time actually observed on each project and the sensitivity of those observations is also saved.

Arguments

projecttable	Name of table to save projects to AllowedString Default:''
allprojects	Save all projects AllowedBool DefaultT
whichprojects	Projects to save (if allprojects==F) AllowedVector of Ints Default[]

Returns
Bool

Example

```
mysched.saveprojects(projecttable='PROJECTS.SAVE')
```

---

dynamicsched.recoverprojects - Function

Package	synthesis
Module	simulator
Tool	dynamicsched

Recover the observing projects from a Table

Synopsis
recoverprojects(projecttable)

Description
Recover the observing projects from a Table, including information about how much time has been observed and the sensitivity. This returns F is the table did not contain a valid project record.

Arguments

projecttable	Name of table to recover projects from Allowed String Default: ''

Returns
Bool

Example

```
mysched.recoverprojects(projecttable='PROJECTS.SAVE')
```

---

dynamicsched.saveschedule - Function

Package	synthesis
Module	simulator
Tool	dynamicsched

Save the current observing schedule to a table

Synopsis
saveschedule(scheduletable)

Description
Save the current observing schedule to a table.

Arguments

scheduletable	Name of table to save the observng schedule to Allowed String Default ''

Returns
Bool

Example

```
mysched.saveschedule(scheduletable='SCHEDULE1.SAVE');
```

---

dynamicsched.recoverschedule - Function

Package	synthesis
Module	simulator
Tool	dynamicsched

Recover an observing schedule from a table

Synopsis
recoverschedule(scheduletable)

Description
Recover an observing schedule from a table. This returns F if the table read in not not contain a valid schedule record.

Arguments

scheduletable	Name of table to recover the observng schedule from Allowed String Default ''

Returns
Bool

Example

```
mysched.recoverschedule(scheduletable='SCHEDULE1.SAVE');
```

---

dynamicsched.defaultinitialize - Function

Package	synthesis
Module	simulator
Tool	dynamicsched

Perform a default initialization -- archaic

Synopsis
defaultinitialize()

Description
Perform a default initialization. This initialization has not tracked various changes in the system and should be used with cation.

Arguments

Returns
Bool

Example

```
mysched.defaultinitialize();
```

---

dynamicsched.done - Function

Package	synthesis
Module	simulator
Tool	dynamicsched

Clean up the dynamic scheduling object

Synopsis
done()

Description
Be done with it already.

Arguments

Returns
Bool

Example

```
mysched.done();
```

---

dynamicsched.getpgplotter - Function

Package	synthesis
Module	simulator
Tool	dynamicsched

Get the pgplotter

Synopsis
getpgplotter()

Description
Setup the pgplotter for plotting.

Arguments

Returns
Bool

Example

```
mysched.getpgplotter();
```

---

dynamicsched.setschedcriteria - Function

Package	synthesis
Module	simulator
Tool	dynamicsched

Set criteria for scheduling observations

Synopsis
setschedcriteria(hatozenith, hafromzenith, phasecutoff)

Description
Set criteria for scheduling observations. Currently, the algorithm used to schedule observations is quite brain-dead, taking sources within some hour angle range of transit, and picking the highest frequency observations that still have residual phase errors less than the cut-off value.

Arguments

hatozenith	Hour angle approaching zenith Allowed:Quantity Default:'2h'
hafromzenith	Hour angle passing zenith Allowed:Quantity Default:'1.5h'
phasecutoff	Maximum residual phase error permitted Allowed:Quantity Default:'30deg'

Returns
Bool

Example

```
mysched.setschedcriteria(hatozenith='1.0h', hafromzenith='1.0h',
phasecutoff='20deg');
```

---

dynamicsched.schedule - Function

Package	synthesis
Module	simulator
Tool	dynamicsched

Go through the site test data and schedule projects

Synopsis
schedule()

Description
Go through the site test data and schedule projects.

Arguments

Returns
Bool

Example

```
mysched.schedule();
```

---

dynamicsched.reobserveschedule - Function

Package	synthesis
Module	simulator
Tool	dynamicsched

Given a schedule reobserve it for the current conditions

Synopsis
reobserveschedule()

Description
A schedule may be obtained through some other method (ie, like recoverschedule), and it can then be reobserved and the sensitivity of the observations can be evaluated.

Arguments

Returns
Bool

Example

```
mysched.reobserveschedule()
```

---

dynamicsched.evaluateobservations - Function

Package	synthesis
Module	simulator
Tool	dynamicsched

Evaluate the observations

Synopsis
evaluateobservations()

Description
Evaluate the observations

Arguments

Returns
Bool

Example

```
mysched.evaluateobservations();
```

1.8 newsimulator - Module

Module for simulation of telescope data
include "newsimulator.g"

Description

newsimulator provides a unified interface for simulation of telescope processing. It can create a MeasurementSet from scratch or read in an existing MeasurementSet, it can predict synthesis data onto the (u,v) coordinates or single dish data onto (ra,dec) points, and it can corrupt this data through Gaussian errors or through specific errors residing in (anti-) calibration tables.
newsimulator **is substantially different from (and better than) simulator. Scripts written for one will not work for the other.**
In the observing phase, newsimulator tries to act like a (simple) telescope. You first make a newsimulator tool, with the name of the MeasurementSet that you wish to construct. Next you use the various set* methods to set up the observing (sources, spectral windows, *etc.*). Each such setup should be given a unique name that will be used in the next step. Then you call the observe method for each observing scan you wish to make. Here you specify the source name, spectral windoow name, and observing times. After this, you have a MeasurementSet that is complete but empty. In the next phase, you fill the MeasurementSet with data from a model and then corrupt the measurements (if desired). To fill it in with a model, use the predict method. Finally, to apply errors, first set up the various effects using the relevant set* methods, and then call corrupt.
Some important points (mostly for the *cognoscenti*):

- One call to observe generates one scan (all rows have the same SCAN_NUMBER).
- The start and stop times specified to observe need not be contiguous and so one can simulate antenna drive times.
- Currently there is no facility for patterns of observing, such as mosaicing, since it is easy to do this via sequences of calls of observe.
- The heavy duty columns (DATA, FLAG, IMAGING_WEIGHT, *etc.* are tiled. New tiles are generated for each scan. Thus the TSM files will not get very large.