The CASA vpmanager tool and the VPManager class

D. Petry (ESO, Garching)

Jan 9, 2012

Abstract

The VPManager class in the synthesis module of CASA code permits CASA imaging and simulation routines to hook up to the vpmanager and let it determine which antenna responses to use for which observatory.

The vpmanager tool (by default "vp" in casapy) is the Python user interface to the VPManager class. It has methods to set up a list of primary beams or voltage patterns (*antenna responses*) and then select in detail which of them is used for which observatory. The distinction of several antenna types for a given observatory (heterogeneous arrays) is supported.

Antenna responses can be selected from either internally hard-coded ones, or response-groups defined via an AntennaResponses table, or user-defined analytic primary beams.

1 Overview

The vpmanager tool is the CASA Python object which constitutes the user interface to the VPManager C++ class. By default it is named "vp" in casapy. This document describes the upgraded version of the tool and class to be included in CASA 3.4.

The VPManager class is implemented as a singleton, i.e. internally there is only one instance at all times. This instance accessed via the static VPManager::Instance() method. It is permanent until casapy is exited or it is reinstantiated via the static VPManager::reset() method.

The vp tool connects to the single instance of VPManager. All settings the user makes with the tool, have effect immediately and are then used by all parts of CASA which access the VPManager class (i.e. eventually all imaging and simulation routines).

In order to enable parallelization, a simple *locking mechanism* protects from simultaneous access to the VPManager. Only the application programmer has to be aware of this. On the tool level the locking is done automatically.

The VPManager instance keeps a simple database of available antenna responses, the *vplist*. This list is initialized at the startup of CASA or by calling the reset() method of the class. In the vp tool, the reset call can be triggered using

```
vp.reset()
```

In order to support heterogeneous interferometer arrays, VPManager permits the use of *antenna types* in addition to observatory or *telescope* names.

For defining a simple response which is only spatially scaled by frequency but otherwise constant, a simple call to the vp tool is sufficient, e.g.:

```
vp.setpbairy(telescope='ALMA',
             dishdiam='12.0m',
             blockagediam='0.75m',
             maxrad='1.784deg',
             reffreq='1.0GHz',
             dopb=True)
```

This will create a new entry in the vplist for an analytic Airy disk antenna response and make it the default response for telescope "ALMA". Subsequent requests to VPManager for a ALMA antenna response will get this Airy disk. Using the `vp.setuserdefault` method, the default can be unset again or changed to a different entry in the vplist.

If whole *response systems* are to be defined for a given telescope, the use of an *AntennaResponses table* is possible. Such a table can be set up using the vp tool method `createantresp()` and then connected to a telescope using a command like

```
vp.setpbantresptable(telescope='ALMA',
                     antresppath=casa['dirs']['data']
                     +'/alma/responses/AntennaResponses-ALMA-RT',
                     dopb=True)
```

where the value of the `antresppath` parameter indicates the path to the AntennaResponses table. Subsequent requests for ALMA antenna responses to VPManager will start a search in the indicated AntennaResponses table for responses matching given parameters. Presently supported search parameters in VPManager::getvp() and vp.getvp() are:

- antenna type

- observation time (used for versioning and for reference frame transformations)
- frequency (as a Measure, the reference frame is respected)
- observing direction (to support elevation and azimuth dependent responses)

An example of a call to vp.getvp() is

```
myrecord = vp.getvp(telescope='ALMA',
                    antennatype = 'DV',
                    obstime = '2009/07/24/10:00:00',
                    freq = 'TOPO 100GHz',
                    obsdirection = 'AZEL 30deg 60deg')
```

If the default antenna response for the given telescope is not defined via an AntennaResponses table, the observation parameters `obstime`, `freq`, and `obsdirection` are not needed and can be omitted. The parameter `antennatype` defaults to empty string. So if no antenna types are distinguished for the given telescope, the simplest call to getvp becomes

```
myrecord = vp.getvp(telescope='HATCREEK')
```

During initialization, VPManager will look for entries in the column "AntennaResponses" of the CASA "Observatories" table. If there are non-blank entries, the string found will be interpreted as the path to the default AntennaResponses table for the given telescope.

Note that the casacore AntennaResponses C++ class (which is used by VPManager to administrate the AntennaResponses tables) also supports the additional search parameters "receiver type" and "beam number". A general interface to the response file name search is available through the vp.getrespimagename() method. But presently this accesses only AntennaResponse tables which are entered as the default table in the Observatories table.

Generally, the vp tool methods provide functionality: to set up new analytic antenna responses, select which antenna responses from the vplist to use for which telescope and antenna type, access the contents of the vplist, create and access an AntennaResponses table, create and load a voltage pattern table.

The latter is achieved with the methods vp.saveastable() and vp.loadfromtable() which can be used to save the vplist and the defaults in a CASA table and reload them at a later time.

In section 4, the methods are described in more detail. The appendix A gives example Python scripts. The appendices B and C show the methods of the VPManager C++

class and give an example of how to use it to get a primary beam in application code. But beforehand, section 2 describes the library of predefined responses available to the vpmanager and section 3 describes the use of a voltage pattern table. Finally, section 5 describes the access to ray traced responses for ALMA.

2 The antenna responses library accessible to the vp tool

2.1 Common antenna responses

Many common voltage pattern (vp) and primary beam (pb) models have been coded into CASA. Currently, the recognized models include DEFAULT, ALMA, ACA, ATCA_L1, ATCA_L2, ATCA_L3, ATCA_S, ATCA_C, ATCA_X, GBT, GMRT, HATCREEK, NRAO12M, NRAO140FT, OVRO, VLA, VLA_INVERSE, VLA_NVSS, VLA_2NULL, VLA_4, VLA_P, VLA_L, VLA_C, VLA_X, VLA_U, VLA_K, VLA_Q, WSRT, and WSRT_LOW. In all cases, the VP/PB model and the beam squint (if present) scale linearly with wavelength. If DEFAULT is selected, the appropriate VP/PB model is selected for the telescope and observing frequency.

2.2 1-D Beam Models

Most beam models are rotationally symmetric (excepting beam squint). From the beam parameterization in terms of the various coefficients and other terms, an internal lookup table with 10000 elements is created for application of the VP/PB to an image.

2.3 Beam Squint

The VP/PB models include beam squint. The VLA_L, VLA_C, VLA_X, VLA_U, VLA_K, and VLA_Q models (which are the defaults for those VLA bands), have the appropriate squint magnitude and orientation, though the orientation has not been verified through processing actual data.

3 The voltage pattern table

In the original design of the vpmanager (before the refactoring in 2011), the only way to communicate with the imaging routines was via the so-called voltage pattern table. This functionality still exists in the present tool. However, it is recommended that all CASA

code now use the VPManager::getvp() methods and access the VPManager::Instance() directly.

A new voltage pattern table can be generated by the vp tool using the vp.saveastable() method. The vp description table can then be read by imager's imager.setvp() method, which instantiates the corresponding voltage patterns from the descriptions and applies them to the images.

```
#
# Lets say we want an Airy Disk voltage pattern for our
# HATCREEK data, but we want to use the system default
# for the OVRO data:
#
vp.setpbairy(telescope='HATCREEK', dopb=T, dishdiam='6.0m',
             blockagediam='0.6m',  maxrad='2arcmin',
             reffreq='100GHz', dosquint=F)
#
vp.setcannedpb(telescope='OVRO', dopb=T, commonpb='DEFAULT', dosquint=F)
#
vp.summarizevps()
#
vp.saveastable(tablename='California.Beaming')
#
```

The voltage pattern table created by vp.saveastable() can also be loaded back into the vpmanager thereby restoring a previous state using the method vp.loadfromtable().

4 The vp tool methods

For an up-to-date reference of the individual method parameters, please use help vp.*method* within casapy.

vpmanager Construct a vpmanager tool (note: the underlying VPManager is a singleton). The vpmanager constructor has no arguments.

saveastable Save the vp or pb descriptions as a table. Each description is in a different row of the table. (bool)

```
--- --- --- --- --- --- Parameters  --- --- --- --- --- ---
  tablename:  Name of table to save vp descriptions in
--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
```

loadfromtable Load the vp or pb descriptions from a table (deleting all previous definitions) (bool)

```
--- --- --- --- --- --- Parameters  --- --- --- --- --- ---
  tablename:  Name of table to load vp descriptions from
--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
```

summarizevps Summarize the currently accumulated VP descriptions to the logger. (bool)

```
--- --- --- --- --- --- Parameters  --- --- --- --- --- ---
  verbose:  Print out full record? Otherwise, print summary. false
--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
```

setcannedpb Select a vp/pb from our library of common pb models If 'DEFAULT' is selected, the system default for that telescope and frequency is used. (record)

```
--- --- --- --- --- --- Parameters  --- --- --- --- --- ---
  telescope:  Which telescope in the MS will use this vp/pb? VLA
  othertelescope:  If telescope=="OTHER", specify name here
  dopb:  Should we apply the vp/pb to this telescope's data? true
  commonpb:  List of common vp/pb models: DEFAULT code figures it out DEFAULT
  dosquint:  Enable the natural beam squint found in the common vp model false
  paincrement:  Increment in Parallactic Angle for asymmetric (i.e., squinted)
                vp application 720deg
  usesymmetricbeam:  Not currently used false
--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
```

setpbairy Make an airy disk vp. Information sufficient to create a portion of the Airy disk voltage pattern. The Airy disk pattern is formed by Fourier transforming a uniformly illuminated aperture and is given by

$$vp_p(i) = (areaRatio*2.0*j_1(x)/x-2.0*j_1(x*lengthRatio)/(x*lengthRatio))/areaNorm, \quad (1)$$

where areaRatio is the dish area divided by the blockage area, lengthRatio is the dish diameter divided by the blockage diameter, and

$$x = \frac{i * maxrad * 7.016 * dishdiam/24.5m}{N_{samples} * 1.566 * 60}. \quad (2)$$

(record)

```
--- --- --- --- --- --- Parameters  --- --- --- --- --- ---
  telescope:  Which telescope in the MS will use this vp/pb? VLA
  othertelescope:  If telescope=="OTHER", specify name here
  dopb:  Should we apply the vp/pb to this telescope's data? true
  dishdiam:  Effective diameter of dish 25.0m
  blockagediam:  Effective diameter of subreflector blockage 2.5m
  maxrad:  Maximum radial extent of the vp/pb (scales with 1/freq) 0.8deg
  reffreq:  Frequency at which maxrad is specified 1.0GHz
  squintdir:  Offset (Measure) of RR beam from pointing center, azel frame
              (scales with 1/freq)
  squintreffreq:  Frequency at which the squint is specified 1.0GHz
  dosquint:  Enable the natural beam squint found in the common vp model false
  paincrement:  Increment in Parallactic Angle for asymmetric (i.e., squinted)
                vp application 720deg
  usesymmetricbeam:  Not currently used false
--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
```

setpbcospoly Make a vp/pb from a polynomial of scaled cosines. A voltage pattern or primary beam of the form

$$VP(x) = \sum_i (coeff_i \cos^{2i}(scale_i x). \tag{3}$$

This is a generalization of the WSRT primary beam model. (record)

```
--- --- --- --- --- --- Parameters  --- --- --- --- --- ---
  telescope:  Which telescope in the MS will use this vp/pb? VLA
  othertelescope:  If telescope=="OTHER", specify name here
  dopb:  Should we apply the vp/pb to this telescope's data? true
  coeff:  Vector of coefficients of cosines -1
  scale:  Vector of scale factors of cosines -1
  maxrad:  Maximum radial extent of the vp/pb (scales with 1/freq) 0.8deg
  reffreq:  Frequency at which maxrad is specified 1.0GHz
  isthispb:  Do these parameters describe a PB or a VP? PB
  squintdir:  Offset (Measure) of RR beam from pointing center, azel frame
              (scales with 1/freq)
  squintreffreq:  Frequency at which the squint is specified 1.0GHz
  dosquint:  Enable the natural beam squint found in the common vp model false
  paincrement:  Increment in Parallactic Angle for asymmetric (i.e., squinted)
                vp application 720deg
```

```
  usesymmetricbeam:  Not currently used false
--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
```

setpbgauss Make a Gaussian vp/pb. Make a Gaussian primary beam given by

$$PB(x) = e^{-(x/(halfwidth* \sqrt{1/\log(2)}))}. \quad (4)$$

(record)

```
--- --- --- --- --- --- Parameters  --- --- --- --- --- ---
  telescope:  Which telescope in the MS will use this vp/pb? VLA
  othertelescope:  If telescope=="OTHER", specify name here
  dopb:  Should we apply the vp/pb to this telescope's data? true
  halfwidth:  Half power half width of the Gaussian at the reffreq 0.5deg
  maxrad:  Maximum radial extent of the vp/pb (scales with 1/freq) 1.0deg
  reffreq:  Frequency at which maxrad is specified 1.0GHz
  isthispb:  Do these parameters describe a PB or a VP? PB
  squintdir:  Offset (Measure) of RR beam from pointing center, azel frame
              (scales with 1/freq)
  squintreffreq:  Frequency at which the squint is specified 1.0GHz
  dosquint:  Enable the natural beam squint found in the common vp model false
  paincrement:  Increment in Parallactic Angle for asymmetric (i.e., squinted)
               vp application 720deg
  usesymmetricbeam:  Not currently used false
--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
```

setpbinvpoly Make a vp/pb as an inverse polynomial. The inverse polynomial describes the inverse of the VP or PB as a polynomial of even powers:

$$1/VP(x) = \sum_i coeff_i * x^{2i}. \quad (5)$$

(record)

```
--- --- --- --- --- --- Parameters  --- --- --- --- --- ---
  telescope:  Which telescope in the MS will use this vp/pb? VLA
  othertelescope:  If telescope=="OTHER", specify name here
  dopb:  Should we apply the vp/pb to this telescope's data? true
  coeff:  Coefficients of even powered terms -1
  maxrad:  Maximum radial extent of the vp/pb (scales with 1/freq) 0.8deg
```

```
  reffreq:  Frequency at which maxrad is specified 1.0GHz
  isthispb:  Do these parameters describe a PB or a VP? PB
  squintdir: Offset (Measure) of RR beam from pointing center, azel frame
             (scales with 1/freq)
  squintreffreq:  Frequency at which the squint is specified 1.0
  dosquint:  Enable the natural beam squint found in the common vp model false
  paincrement: Increment in Parallactic Angle for asymmetric (i.e., squinted)
               vp application 720deg
      usesymmetricbeam:  Not currently used false
--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
```

setpbnumeric Make a vp/pb from a user-supplied vector. Supply a vector of vp/pb sample values taken on a regular grid between x=0 and x=maxrad. We perform sinc interpolation to fill in the lookup table.

```
--- --- --- --- --- --- Parameters  --- --- --- --- --- ---
  telescope:  Which telescope in the MS will use this vp/pb? VLA
  othertelescope:  If telescope=="OTHER", specify name here
  dopb:  Should we apply the vp/pb to this telescope's data? true
  vect:  Vector of vp/pb samples uniformly spaced from 0 to maxrad -1
  maxrad:  Maximum radial extent of the vp/pb (scales with 1/freq) 0.8deg
  reffreq:  Frequency at which maxrad is specified 1.0GHz
  isthispb:  Do these parameters describe a PB or a VP? PB
  squintdir:  Offset (Measure) of RR beam from pointing center, azel frame
              (scales with 1/freq)
  squintreffreq:  Frequency at which the squint is specified 1.0GHz
  dosquint:  Enable the natural beam squint found in the common vp model false
  paincrement:  Increment in Parallactic Angle for asymmetric (i.e., squinted)
                vp application 720deg
  usesymmetricbeam:  Not currently used false
--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
```

setpbimage Make a vp/pb from a user-supplied image

Supply an image of the E Jones elements. The format of the image is:

Shape nx by ny by 4 complex polarizations (RR, RL, LR, LL or XX, XY, YX, YY) by 1 channel.

Direction coordinate Az, El

Stokes coordinate All four "stokes" parameters must be present in the sequence RR, RL, LR, LL or XX, XY, YX, YY.

Frequency Only one channel is currently needed - frequency dependence beyond that is ignored.

If a complexiimage is specified the real and imaginary images is to be left empty.

The other option is to provide the real and imaginary part of the E-Jones as separable `float` images On that case one or two images may be specified - the real (must be present) and imaginary parts (optional).

Note that beamsquint must be intrinsic to the images themselves. This will be accounted for correctly by regridding of the images from Az-El to Ra-Dec according to the parallactic angle. (record)

```
--- --- --- --- --- --- Parameters  --- --- --- --- --- ---
  telescope:  Which telescope in the MS will use this vp/pb? VLA
  othertelescope:  If telescope=="OTHER", specify name here
  dopb:  Should we apply the vp/pb to this telescope's data? true
  realimage:  Real part of vp as an image
  imagimage:  Imaginary part of vp as an image
  complexiimage:  complex vp as an image of complex numbers;
                  if specified realimage and imagimage are ignored
--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
```

setpbpoly Make a vp/pb from a polynomial. The VP or PB is described as a polynomial of even powers:

$$VP(x) = \sum_i coeff_i * x^{2i}. \tag{6}$$

(record)

```
--- --- --- --- --- --- Parameters  --- --- --- --- --- ---
  telescope:  Which telescope in the MS will use this vp/pb? VLA
  othertelescope:  If telescope=="OTHER", specify name here
  dopb:  Should we apply the vp/pb to this telescope's data? true
  coeff:  Coefficients of even powered terms -1
  maxrad:  Maximum radial extent of the vp/pb (scales with 1/freq)
           0.8deg
  reffreq:  Frequency at which maxrad is specified 1.0GHz
```

```
  isthispb:  Do these parameters describe a PB or a VP? PB
  squintdir:  Offset (Measure) of RR beam from pointing center,
              azel frame (scales with 1/freq)
  squintreffreq:  Frequency at which the squint is specified 1.0GHz
  dosquint:  Enable the natural beam squint found in the common vp model
             false
  paincrement:  Increment in Parallactic Angle for asymmetric
                (i.e., squinted) vp application 720
  usesymmetricbeam:  Not currently used false
--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
```

setpbantresptable Declare a reference to an antenna responses table. Declare a reference to an antenna responses table containing a set of VP/PB definitions. (bool)

```
--- --- --- --- --- --- Parameters  --- --- --- --- --- ---
  telescope:  Which telescope in the MS will use this vp/pb?
  othertelescope:  If telescope=="OTHER", specify name here
  dopb:  Should we apply the vp/pb to this telescope's data? true
  antresppath:  The path to the antenna responses table
                (absolute or relative to CASA data dir.)
--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
```

reset Reinitialize the VPManager. This will erase the vplist and defaults defined on the command line. During initialization, VPManager will look for entries in the column "AntennaResponses" of the CASA "Observatories" table. If there are non-blank entries, the string found will be interpreted as the path to the default AntennaResponses table for the given telescope. (bool)

setuserdefault Select the VP which is to be used for the given telescope and antenna type. Overwrites a previous default. There can be one global default for each telescope and one specific default for each (telescope, antennatype) pair. The global default will be used when no antenna type is given or no specific default for the (telescope, antennatype) pair exists. A vplistnum=-2 will unset an existing default for the (telescope, antennatype) pair.(bool)

```
--- --- --- --- --- --- Parameters  --- --- --- --- --- ---
  vplistnum:  The number of the vp as displayed by summarizevps()
              or -1 for internal or -2 for unset, default -1
  telescope:  Which telescope in the MS will use this vp/pb?
  anttype:  Which antennatype will use this vp/pb? Default: "" = all
--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
```

getuserdefault Get the vp list number of the present default VP/PB for the given parameters. A return value of −1 means that the common library default PB for the telescope is presently the default. (int)

```
--- --- --- --- --- --- Parameters  --- --- --- --- --- ---
  telescope:  Which telescope in the MS will use this vp/pb?
  anttype:  Which antennatype will use this vp/pb? Default: "" = all
--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
```

getanttypes Return the list of available antenna types for the given telescope, antennatype and observation parameters. (string array)

```
--- --- --- --- --- --- Parameters  --- --- --- --- --- ---
  telescope:  Telescope name
  obstime:  Time of the observation
            (for versioning and reference frame calculations)
  freq:  Frequency of the observation
         (may include reference frame, default: LSRK)
  obsdirection:  Direction of the observation
                 (may include reference frame, default: J2000).
                 default: Zenith =  AZEL 0deg 90deg
--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
```

numvps Return the number of vps/pbs available for the given parameters. Can be used to determine the number of antenna types. Note: if a global response is defined for the telescope, this will increase the count of available vps/pbs by 1. (int)

```
--- --- --- --- --- --- Parameters  --- --- --- --- --- ---
  telescope:  Telescope name
  obstime:  Time of the observation
               (for versioning and reference frame calculations)
  freq:  Frequency of the observation
         (may include reference frame, default: LSRK)
  obsdirection:  Direction of the observation
                 (may include reference frame, default: J2000).
                 default: Zenith = AZEL 0deg 90deg
--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
```

getvp Return the default vps/pbs *record* for the given parameters. Record is empty if no matching vp/pb could be found. (record)

```
--- --- --- --- --- --- Parameters  --- --- --- --- --- ---
  telescope:  Telescope name
  obstime:  Time of the observation
            (for versioning and reference frame calculations),
            e.g. 2011/12/12T00:00:00
  freq:  Frequency of the observation
             (may include reference frame, default: LSRK)
  antennatype:  The antenna type as a string, e.g. "DV"
  obsdirection:  Direction of the observation
                 (may include reference frame, default: J2000),
                  default: AZEL 0deg 90deg
--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
```

createantresp Create a standard-format AntennaResponses table. (bool)

```
--- --- --- --- --- --- Parameters  --- --- --- --- --- ---
  imdir:  Path to the directory containing the response images
  starttime:  Time from which onwards the response is valid,
              format YYYY/MM/DD/hh:mm:ss
  bandnames:  List containing the names of the observatory's frequency bands
  bandminfreq:  List containing the lower edges of the observatory's
                frequency bands, e.g. ["80GHz","120GHz"]
  bandmaxfreq:  List containing the upper edges of the observatory's
                frequency bands, e.g. ["120GHz","180GHz"]
--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
```

The AntennaResponses table serves CASA to look up the location of images describing the response of observatory antennas. Three types of images are supported: "VP" - real voltage patterns, "AIF" - complex aperture illumination patterns, "EFP" - complex electric field patterns. For each image, a validity range can be defined in Azimuth, Elevation, and Frequency. Furthermore, an antenna type (for heterogeneous arrays), a receiver type (for the case of several receivers on the same antenna having overlapping frequency bands), and a beam number (for the case of multiple beams per antenna) are associated with each response image.

The images need to be stored in a single directory DIR of arbitrary name and need to have file names following the pattern

```
obsname_beamnum_anttype_rectype_azmin_aznom_azmax_elmin_elnom_elmax\
_freqmin_freqnom_freqmax_frequnit_comment_functype.im
```

where the individual name elements mean the following (none of the elements may contain the space character, but they may be empty strings if they are not numerical values):

obsname - name of the observatory as in the Observatories table, e.g. ”ALMA”

beamnum - the numerical beam number (integer) for the case of multiple beams, e.g. 0

anttype - name of the antenna type, e.g. ”DV”

rectype - name of the receiver type, e.g. ””

azmin, aznom, azmax - numerical value (degrees) of the minimal, the nominal, and the maximal Azimuth where this response is valid, e.g. ”-10.5_0._10.5”

elmin, elnom, elmax - numerical value (degrees) of the minimal, the nominal, and the maximal Elevation where this response is valid, e.g. ”10._45._80.”

freqmin, freqnom, freqmax - numerical value (degrees) of the minimal, the nominal, and the maximal Frequency (in units of frequnit) where this response is valid, e.g. ”84._100._116.”

frequnit - the unit of the previous three frequencies, e.g. ”GHz”

comment - any string containing only characters permitted in file names and not empty space

functype - the type of the image as defined above (”VP”, ”AIF”, or ”EFP”)

The createantresp method will then extract the parameters from all the images in DIR and create the lookup table in the same directory.

getrespimagename Given the observatory name, the antenna type, the receiver type, the observing frequency, the observing direction, and the beam number, find the applicable response image and return its name. (string)

```
--- --- --- --- --- --- Parameters  --- --- --- --- --- ---
  telescope:  Which telescope is described by this response?
  starttime:  Time at which the response has to be valid,
              format YYYY/MM/DD/hh:mm:ss
  frequency:  The frequency at which the response has to be valid,
              e.g. "100GHz"
  functype:  Type of the responsefunction requested, e.g. "EFP" ANY
  anttype:  Antenna type (observatory-dependent)
```

```
    azimuth:  Azimuth of the observation
              (at the location of the observatory, 0 is North),
              e.g. "5deg" 0deg
    elevation:  Elevation of the observation
              (at the location of the observatory, 0 is North),
              e.g. "60deg" 45deg
    rectype:  Receiver type (observatory-dependent)
    beamnumber:  Beam number (for the case of multiple beams per receiver) 0
--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
```

5 Access to ray traced responses for ALMA

The AntennaResponses table class supports the responses function type INTERNAL which means that the responses are generated by CASA using Walter Brisken's ray tracing code.

The VPManager also supports this type, however, only for the telescopes ALMA, ACA, and OSF. This happens via the class ALMACalcIlluminationConvFunc in VPManager::getvp().

VPManager::getvp() will generate ray traced response images and store them in the current working directory under standard names with the format

```
BeamCalcTmpImage_<telescope>_<antennatype>_<frequency>MHz
```

The images are reused if they already exist. If the user wants to regenerate them, they first need to be deleted.

A vp tool usage examples

A.1 Define Airy beams for ALMA antenna types

```
vp.reset()
vp.setpbairy(telescope='ALMA',
             dishdiam='11m',
             blockagediam='0.75m',
             maxrad='1.784deg',
             reffreq='1.0GHz',
             dopb=True)
myid1 = vp.getuserdefault('ALMA')

vp.setpbairy(telescope='ALMA',
             dishdiam='6m',
             blockagediam='0.75m',
             maxrad='3.5deg',
             reffreq='1.0GHz',
             dopb=True)
myid2 = vp.getuserdefault('ALMA')

vp.setuserdefault(myid1, 'ALMA', 'DV')
vp.setuserdefault(myid1, 'ALMA', 'DA')
vp.setuserdefault(myid1, 'ALMA', 'PM')
vp.setuserdefault(myid2, 'ALMA', 'CM')
# unset the global default for ALMA such that only the explicitly
# defined antenna types are valid
vp.setuserdefault(-2, 'ALMA', '')
```

A.2 Define a reference to an AntennaResponses table for ALMA

```
vp.setpbantresptable(telescope='ALMA',
                     antresppath=casa['dirs']['data']
                     +'/alma/responses/AntennaResponses-ALMA-RT',
                     dopb=True)
```

B VPManager class public methods

```
// this is a SINGLETON class
static VPManager* Instance();
static void reset();

// call before anything else (returns False if already locked)
Bool lock();
// or if you are ready to wait for the lock (returns True if successful)
Bool acquireLock(Double timeoutSecs,
                 Bool verbose=False);
// verbose check for lock
Bool isLocked();
// call when you are done
void release();

Bool saveastable(const String& tablename);

Bool loadfromtable(const String& tablename);

Bool summarizevps(const Bool verbose);


Bool setcannedpb(const String& tel,
                 const String& other,
                 const Bool dopb,
                 const String& commonpb,
                 const Bool dosquint,
                 const Quantity& paincrement,
                 const Bool usesymmetricbeam,
                 Record& rec);

Bool setpbairy(const String& telescope, const String& othertelescope,
               const Bool dopb, const Quantity& dishdiam,
               const Quantity& blockagediam,
               const Quantity& maxrad,
               const Quantity& reffreq,
               MDirection& squintdir,
               const Quantity& squintreffreq, const Bool dosquint,
               const Quantity& paincrement,
```

```
                const Bool usesymmetricbeam,
                Record& rec);

Bool setpbcospoly(const String& telescope, const String& othertelescope,
                  const Bool dopb, const Vector<Double>& coeff,
                  const Vector<Double>& scale,
                  const Quantity& maxrad,
                  const Quantity& reffreq,
                  const String& isthispb,
                  MDirection& squintdir,
                  const Quantity& squintreffreq, const Bool dosquint,
                  const Quantity& paincrement,
                  const Bool usesymmetricbeam,
                  Record& rec);

Bool setpbgauss(const String& tel, const String& other, const Bool dopb,
                const Quantity& halfwidth, const Quantity maxrad,
                const Quantity& reffreq, const String& isthispb,
                MDirection& squintdir, const Quantity& squintreffreq,
                const Bool dosquint, const Quantity& paincrement,
                const Bool usesymmetricbeam, Record& rec);

Bool setpbinvpoly(const String& telescope, const String& othertelescope,
                  const Bool dopb, const Vector<Double>& coeff,
                  const Quantity& maxrad,
                  const Quantity& reffreq,
                  const String& isthispb,
                  MDirection& squintdir,
                  const Quantity& squintreffreq, const Bool dosquint,
                  const Quantity& paincrement,
                  const Bool usesymmetricbeam,
                  Record& rec);

Bool setpbnumeric(const String& telescope, const String& othertelescope,
                  const Bool dopb, const Vector<Double>& vect,
                  const Quantity& maxrad,
                  const Quantity& reffreq,
                  const String& isthispb,
                  MDirection& squintdir,
                  const Quantity& squintreffreq, const Bool dosquint,
                  const Quantity& paincrement,
```

```
                  const Bool usesymmetricbeam,
                  Record &rec);

Bool setpbimage(const String& telescope, const String& othertelescope,
                const Bool dopb, const String& realimage,
                const String& imagimage, const String& compleximage,
                Record& rec);

Bool setpbpoly(const String& telescope, const String& othertelescope,
               const Bool dopb, const Vector<Double>& coeff,
               const Quantity& maxrad,
               const Quantity& reffreq,
               const String& isthispb,
               MDirection& squintdir,
               const Quantity& squintreffreq, const Bool dosquint,
               const Quantity& paincrement,
               const Bool usesymmetricbeam,
               Record &rec);


Bool setpbantresptable(const String& telescope, const String& othertelescope,
                       const Bool dopb, const String& tablepath);
                      // no record filled, need to access via getvp()

// set the default voltage pattern for the given telescope
Bool setuserdefault(const Int vplistfield,
                    const String& telescope,
                    const String& antennatype="");

Bool getuserdefault(Int& vplistfield,
                    const String& telescope,
                    const String& antennatype="");

Bool getanttypes(Vector<String>& anttypes,
                 const String& telescope,
                 const MEpoch& obstime,
                 const MFrequency& freq,
                 const MDirection& obsdirection);
```

```
// return number of voltage patterns satisfying the given constraints
Int numvps(const String& telescope,
           const MEpoch& obstime,
           const MFrequency& freq,
           const MDirection& obsdirection=MDirection(Quantity( 0., "deg"),
                                                     Quantity(90., "deg"),
                                                     MDirection::AZEL)
           );

// get the voltage pattern satisfying the given constraints
Bool getvp(Record &rec,
           const String& telescope,
           const MEpoch& obstime,
           const MFrequency& freq,
           const String& antennatype="",
           const MDirection& obsdirection=MDirection(Quantity( 0., "deg"),
                                                     Quantity(90., "deg"),
                                                     MDirection::AZEL)
           );

// get a general voltage pattern for the given telescope and ant type if available
Bool getvp(Record &rec,
           const String& telescope,
           const String& antennatype=""
           );
```

C VPManager usage examples - access from C++ code

Example of how to obtain a PBMath object for a given observation.

```
PBMath* myPBp = 0;

String telescope;
String antennatype;
MEpoch mObsTime;
MFrequency mFreq;
MDirection mObsDir;

// insert code here to fill telescope, antennatype, mObsTime, mFreq, and  mObsDir

Record rec;

// before the first access to the VPManager, the routine has to acquire the lock
if(!VPManager::Instance()->acquireLock(60, True)){ // e.g. 60 seconds timeout, verbose
    // acquireLock produces its own error message in the logger when verbose
    return False;
}

if(!VPManager::Instance()->getvp(rec,
                                 telescope, mObsTime, mFreq, antennatype, mObsDir)){
    os << LogIO::SEVERE << "Could not obtain PB from vpmanager." << LogIO::POST;
    // release the lock when you are done!
    VPManager::Instance()->release();
    return False;
}

// if the above call was your last one to VPManager, release the lock
VPManager::Instance()->release();

myPBp = new PBMath(rec);
```

If the observation parameters time, frequency, and observation direction are not known at the time of the call, the simplified overloaded version of the getvp() method can be used.

```
PBMath* myPBp = 0;

String telescope;
String antennatype;

// insert code here to fill telescope and antennatype

Record rec;

if(!VPManager::Instance()->acquireLock(60, True)){
    return False;
}

if(!VPManager::Instance()->getvp(rec,telescope, antennatype)){
    os << LogIO::SEVERE << "Could not obtain PB from vpmanager." << LogIO::POST;
    VPManager::Instance()->release();
    return False;
}

// if the above call was your last one to VPManager, release the lock
VPManager::Instance()->release();

myPBp = new PBMath(rec);
```

This works if the user has not set the default beam to point to an AntennaResponses table. If an AntennaResponses table has to be accessed, the above call to getvp() will give an appropriate error message and return False.

A Vector of the unique antenna types can be obtained with the getanttypes() method:

```
String telescope;

MEpoch mObsTime;
MFrequency mFreq;
MDirection mObsDir;

Bool rval;

Vector<String> antTypes;

if(!VPManager::Instance()->acquireLock(60, True)){
    return False;
}

rval = VPManager::Instance()->getanttypes(antTypes,
                                          telescope, mObsTime, mFreq, mObsDir);
VPManager::Instance()->release();

if(!rval){
    os << LogIO::SEVERE << "No responses available for telescope "
       << telescope << LogIO::POST;
    return False;
}
```

This will set the Vector of Strings `antTypes` to contain one entry for each valid antenna type.

NOTE:If there is a global response for the telescope (i.e. one without a specific antenna type), the Vector will contain an element which is an empty String. So if there is a global response, the size of the antTypes Vector in the example above will be 1 + the number of specific antenna types.

If there is no response at all for the given parameters, getanttypes() will return False and the Vector will be empty.

The method numvps() will internally call getanttypes() and return the size of the antTypes Vector.