Monday Jan 11
900 coffee, light breakfast, introductions & logistics [RI]
- updates: casaguides.nrao.edu and one sim example [RI]
- DSRP++ examples library wiki [vanKampen] (begin filling in, Eelco can present to NAASC Tues)
- the guts of simdata – the task interfacing with the tools [RI]
- discussion – improvements to simdata interface and outputs
(1000 regular CASA developer telecon)
- (CASA developers) uses/requirements for simulation as used by CASA developers and pipeline group
1050 uses/requirements for simulation as used by CSV [Corder]
1110 web interface (existing AIPS+parseltongue, concept for CASA) [Heywood]
discussion – write requirements for new web interface
1210 Corder lunch talk
110 logistics for implementing web interface [Halstead & NRAO IT]
215 how Simulator, NewMSSimulator, and Imager calculate visibilities
315 SD1 – update on sdsim, requirements to produce coordinated SD and interferometric ms for e.g. pipeline testing
how pointing corruption and correction work in CASA [Bhatnagar]

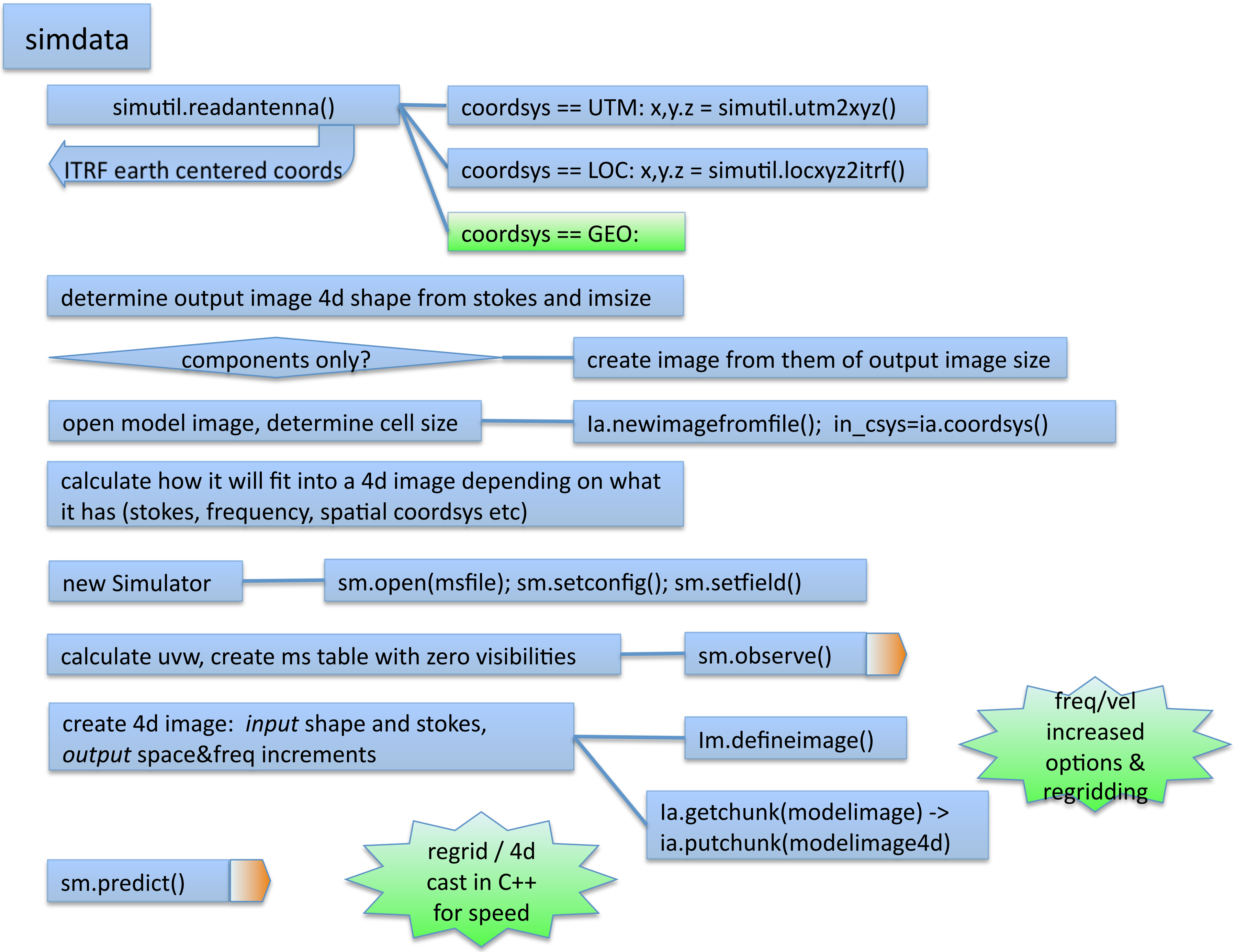
Tuesday Jan 12
900 coffee, light breakfast
910 CASA calibration with the Measurement Equation and VisCals
how Simulator instructs VisCals to invent themselves,
- aatm, the WVR methods in particular and how to best simulate WVR measurements
- any required changes to Calibrator and VisCals to tie those in with the others? [+Moellenbrock]
- ASDM structure, issues regarding simulating perfect visibilities and their corrupting cal in an ASDM
1210 Nikolic lunch talk
115 items of interest for general NAASC staff (there is a regular NAASC internal meeting at this time). In particular, are the items on the library wiki/list appropriate, and who in the room wants to sign up for one?
130 discussion: what will (should) the sum of prep tools look to the user?
simdata + OT + etime calculator + helpdesk + web interface
245 incorporation of current and current and future simulations in the archive [vanKampen, Lacy]
300 implement the above ☺

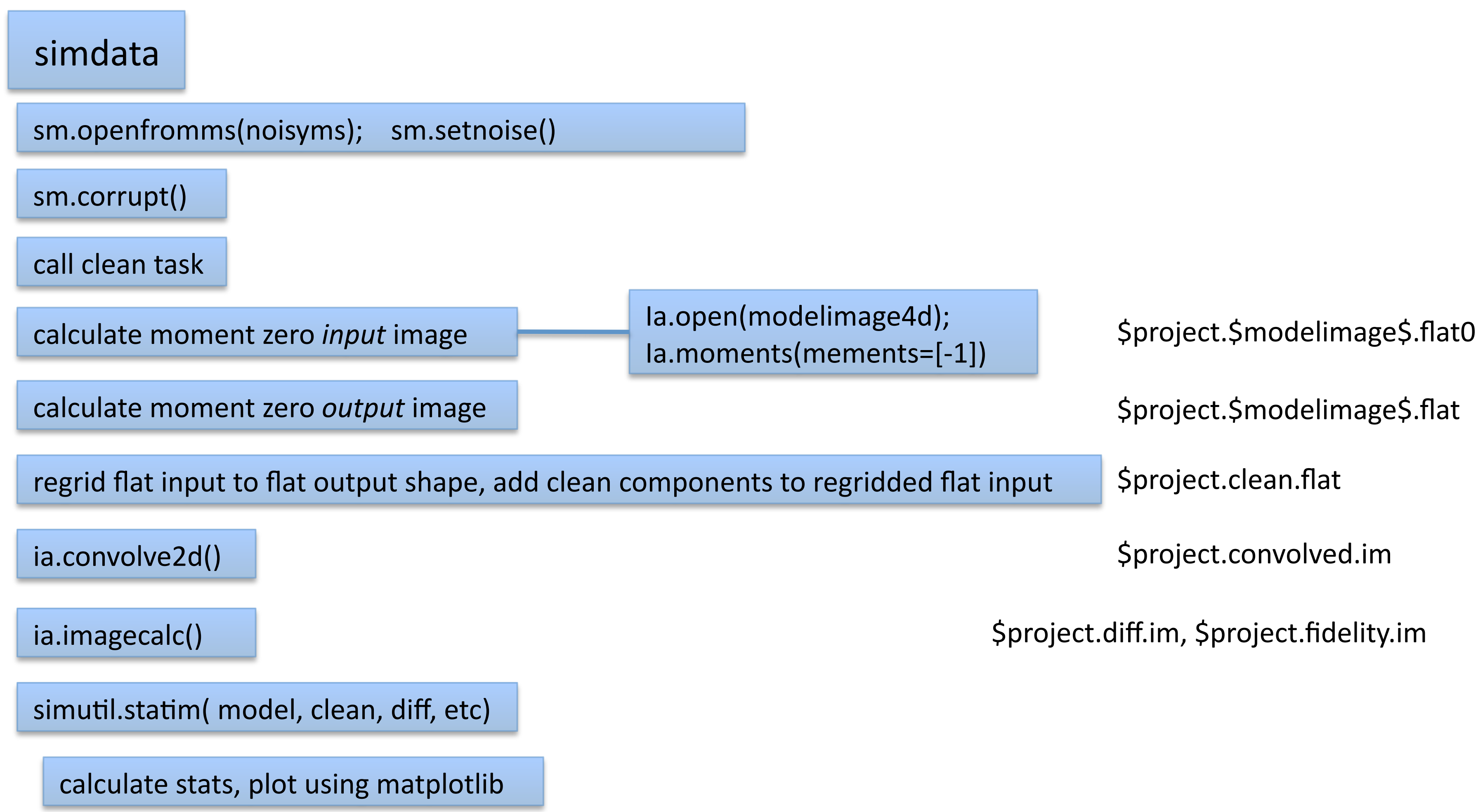
python

C++

Stub/unimplemented code

Places for upgrades

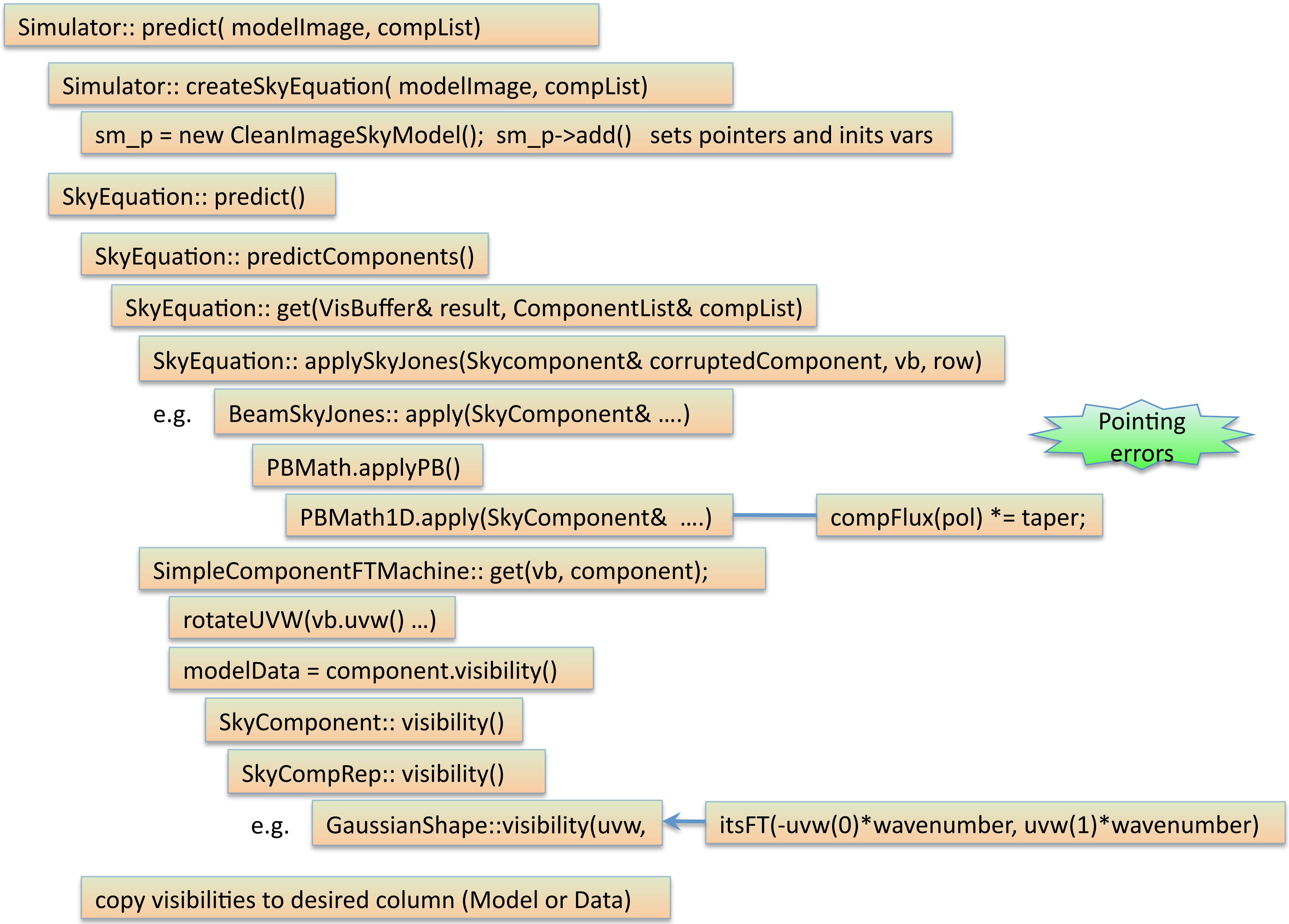

simdata
simutil.readantenna()
ITRF earth centered coords
coordsys == UTM: x,y.z = simutil.utm2xyz()
coordsys == LOC: x,y.z = simutil.locxyz2itrf()
coordsys == GEO:
determine output image 4d shape from stokes and imsize
components only?
create image from them of output image size
open model image, determine cell size
Ia.newimagefromfile(); in_csys=ia.coordsys()
calculate how it will fit into a 4d image depending on what it has (stokes, frequency, spatial coordsys etc)
new Simulator
sm.open(msfile); sm.setconfig(); sm.setfield()
calculate uvw, create ms table with zero visibilities
sm.observe()
create 4d image: *input* shape and stokes, *output* space&freq increments
Im.defineimage()
Ia.getchunk(modelimage) -> ia.putchunk(modelimage4d)
freq/vel increased options & regridding
sm.predict()
regrid / 4d cast in C++ for speed


simdata
sm.openfromms(noisyms);    sm.setnoise()
sm.corrupt()
call clean task
calculate moment zero *input* image
Ia.open(modelimage4d);
Ia.moments(mements=[-1])
$project.$modelimage$.flat0
calculate moment zero *output* image
$project.$modelimage$.flat
regrid flat input to flat output shape, add clean components to regridded flat input
$project.clean.flat
ia.convolve2d()
$project.convolved.im
ia.imagecalc()
$project.diff.im, $project.fidelity.im
simutil.statim( model, clean, diff, etc)
calculate stats, plot using matplotlib

Simulator::observe(source,spw,startTime,stopTime)

- NewMSSimulator::observe(source,spw,startTime,stopTime)
- get antXYZ from antenna subtable, feed, spw, source info from their subtables
- add nIntegrations rows and extend hypercube and subtables
- NewMSSimulator::calcAntUVW()
- Put UVW values in new rows — If autoCorrelationWt_p > 0 anslo add AC rows
- flag based on elevation and shadowing
- add exact phase center to Pointing subtable

Pointing errors?



Simulator:: predict( modelImage, compList)
Simulator:: createSkyEquation( modelImage, compList)
sm_p = new CleanImageSkyModel(); sm_p->add() sets pointers and inits vars
SkyEquation:: predict()
SkyEquation:: predictComponents()
SkyEquation:: get(VisBuffer& result, ComponentList& compList)
SkyEquation:: applySkyJones(Skycomponent& corruptedComponent, vb, row)
e.g.
BeamSkyJones:: apply(SkyComponent& ….)
Pointing errors
PBMath.applyPB()
PBMath1D.apply(SkyComponent& ….)
compFlux(pol) *= taper;
SimpleComponentFTMachine:: get(vb, component);
rotateUVW(vb.uvw() …)
modelData = component.visibility()
SkyComponent:: visibility()
SkyCompRep:: visibility()
e.g.
GaussianShape::visibility(uvw,
itsFT(-uvw(0)*wavenumber, uvw(1)*wavenumber)
copy visibilities to desired column (Model or Data)

- Simulator:: predict( modelImage, compList)
  - Simulator:: createSkyEquation( modelImage, compList)
    - sm_p = new CleanImageSkyModel(); sm_p->add() sets pointers and inits vars
  - SkyEquation:: predict()
    - SkyEquation:: predictComponents()
    - copy visibilities to desired column of VB (Model or Data)
    - SkyEquation:: initalizeGet()
      - SkyEquation:: applySkyJones(vb, row, ImageInterface&
        - e.g. BeamSkyJones:: apply()
          - PBMath.applyPB()
    - SkyEquation:: get(VisBuffer& result, Int model)
      - e.g. GridFT::get(
        - rotateUVW(vb.uvw …; refocus(vb.uvw ….
        - FTMachine::getInterpolateArrays(
        - fgridft.f : dgrid()
    - copy visibilities to desired column of VB (Model or Data)
  - copy visibilities from VB back to VI

Apply atmosphere TJones here?

Pointing errors

Stolen from George Moellenbrock's Synthesis Summer School lecture

# From Idealistic to Realistic

- Formally, we wish to use our interferometer to obtain the visibility function, which we intend to invert to obtain an image of the sky:

$$V(u,v) = \int_{sky} I(l,m) e^{-i2\pi(ul+vm)} dl\, dm$$

- In practice, we correlate (multiply & average) the electric field (voltage) samples, $x_i$ & $x_j$, received at pairs of telescopes (*i,j*) and processed through the observing system:

$$V_{ij}^{obs}(u_{ij}, v_{ij}) = \left\langle x_i(t) \cdot x_j^*(t) \right\rangle_{\Delta t} = J_{ij} V(u_{ij}, v_{ij})$$

  - Averaging duration is set by the expected timescales for variation of the correlation result (typically 10s or less for the VLA)
- $J_{ij}$ is an *operator* characterizing the net effect of the observing process for baseline *(i,j),* which we must *calibrate*
- Sometimes $J_{ij}$ corrupts the measurement irrevocably, resulting in data that must be *edited*

Stolen from George Moellenbrock's Synthesis Summer School lecture

# Antenna-based Cross Calibration

- Measured visibilities are formed from a product of antenna-based signals. Can we take advantage of this fact?
- The net signal delivered by antenna *i*, $x_i(t)$, is a combination of the desired signal, $s_i(t,l,m)$, corrupted by a factor $J_i(t,l,m)$ and integrated over the sky, and diluted by noise, $n_i(t)$:

$$x_i(t) = \int_{sky} J_i(t,l,m) s_i(t,l,m)\, dl dm + n_i(t)$$

$$= s_i'(t) + n_i(t)$$

- $J_i(t,l,m)$ is the product of a series of effects encountered by the incoming signal
- $J_i(t,l,m)$ is an *antenna-based* complex number
- Usually, $|n_i| >> |s_i|$

Stolen from George Moellenbrock's Synthesis Summer School lecture

# Correlation of Realistic Signals - I

- The correlation of two realistic signals from different antennas:

$$\left\langle x_i \cdot x_j^* \right\rangle_{\Delta t} = \left\langle (s_i' + n_i) \cdot (s_j' + n_j)^* \right\rangle_{\Delta t}$$

$$= \left\langle s_i' \cdot s_j'^* \right\rangle + \left\langle s_i' \cdot n_j^* \right\rangle + \left\langle n_i \cdot s_j'^* \right\rangle + \left\langle n_i \cdot n_j^* \right\rangle$$

- Noise signal doesn't correlate—even if $|n_i| >> |s_i|$, the correlation process isolates desired signals:

$$= \left\langle s_i' \cdot s_j'^* \right\rangle_{\Delta t}$$

$$= \left\langle \int_{sky} J_i s_i \, dl' dm' \cdot \int_{sky} J_j^* s_j^* \, dl dm \right\rangle_{\Delta t}$$

- In integral, only $s_i(t,l,m)$, from the same directions correlate (i.e., when $l=l'$, $m=m'$), so order of integration and signal product can be reversed:

$$= \left\langle \int_{sky} J_i J_j^* s_i s_j^* \, dl dm \right\rangle_{\Delta t}$$

Stolen from George Moellenbrock's Synthesis Summer School lecture

# Correlation of Realistic Signals - II



- The $s_i$ & $s_j$ differ *only* by the relative arrival phase of signals from different parts of the sky, yielding the Fourier phase term (to a good approximation):

$$V_{ij} = \left\langle \int_{sky} J_i J_j^* s^2(t,l,m) e^{-i2\pi(u_{ij}l + v_{ij}m)} dldm \right\rangle_{\Delta t}$$

- On the timescale of the averaging, the only meaningful average is of the *squared* signal itself (direction-dependent), which is just the image of the source:

$$= \int_{sky} J_i J_j^* \left\langle s^2(t,l,m) \right\rangle_{\Delta t} e^{-i2\pi(u_{ij}l + v_{ij}m)} dldm$$

$$= \int_{sky} J_i J_j^* I(l,m) e^{-i2\pi(u_{ij}l + v_{ij}m)} dldm$$

- If all *J=1,* we of course recover the ideal expression:

$$= \int_{sky} I(l,m) e^{-i2\pi(u_{ij}l + v_{ij}m)} dldm$$

Stolen from George Moellenbrock's Synthesis Summer School lecture

# The Scalar Measurement Equation



$$V_{ij}^{obs} = \int_{sky} J_i J_j^* I(l,m) e^{-i2\pi(u_{ij}l + v_{ij}m)} dl dm$$

- First, isolate non-direction-dependent effects, and factor them from the integral:

$$= \left(J_i^{vis} J_j^{vis*}\right) \int_{sky} \left(J_i^{sky} J_j^{sky*}\right) I(l,m) e^{-i2\pi(u_{ij}l + v_{ij}m)} dl dm$$

- Next, we recognize that over small fields of view, it is possible to assume $J^{sky}=1$, and we have a relationship between ideal and observed Visibilities:

$$= \left(J_i^{vis} J_j^{vis*}\right) \int_{sky} I(l,m) e^{-i2\pi(u_{ij}l + v_{ij}m)} dl dm$$

$$V_{ij}^{obs} = \left(J_i^{vis} J_j^{vis*}\right) V_{ij}^{true} = J_i J_j^* V_{ij}^{true}$$

- Standard calibration of most existing arrays reduces to solving this last equation for the $J_i$

Stolen from George Moellenbrock's Synthesis Summer School lecture

# Solving for the $J_i$



- We can write: $$\frac{V_{ij}^{obs}}{V_{ij}^{true}} - \left(J_i J_j^*\right) = 0$$
- …and define chi-squared: $$\chi^2 = \sum_{\substack{i,j \\ i \neq j}} \left| \frac{V_{ij}^{obs}}{V_{ij}^{true}} - \left(J_i J_j^*\right) \right|^2 w_{ij}$$
- …and minimize chi-squared w.r.t. each *Ji*, yielding (iteration):

$$J_i = \sum_{\substack{j \\ j \neq i}} \left( \frac{V_{ij}^{obs}}{V_{ij}^{true}} J_j w_{ij} \right) \Big/ \sum_{\substack{j \\ j \neq i}} \left( \left|J_j\right|^2 w_{ij} \right) \qquad \left( \frac{\partial \chi^2}{\partial J_i^*} = 0 \right)$$

- …which we recognize as a weighted average of $J_i$, itself:

$$J_i = \sum_{\substack{j \\ j \neq i}} \left( J_i' w_{ij}' \right) \Big/ \sum_{\substack{j \\ j \neq i}} w_{ij}'$$

Stolen from George Moellenbrock's Synthesis Summer School lecture

# Full-Polarization Formalism: Signal Domain

- Substitute:

$$s_i \rightarrow \vec{s}_i = \begin{pmatrix} s^p \\ s^q \end{pmatrix}_i, \quad J_i \rightarrow \overleftrightarrow{J}_i = \begin{pmatrix} J^{p\rightarrow p} & J^{q\rightarrow p} \\ J^{p\rightarrow q} & J^{q\rightarrow q} \end{pmatrix}$$

- The *Jones matrix* thus corrupts the vector wavefront signal as follows:

$$\vec{s}_i' = \overleftrightarrow{J}_i \vec{s}_i \qquad \text{(sky integral omitted)}$$

$$\begin{pmatrix} s'^p \\ s'^q \end{pmatrix}_i = \begin{pmatrix} J^{p\rightarrow p} & J^{q\rightarrow p} \\ J^{p\rightarrow q} & J^{q\rightarrow q} \end{pmatrix}_i \begin{pmatrix} s^p \\ s^q \end{pmatrix}_i$$

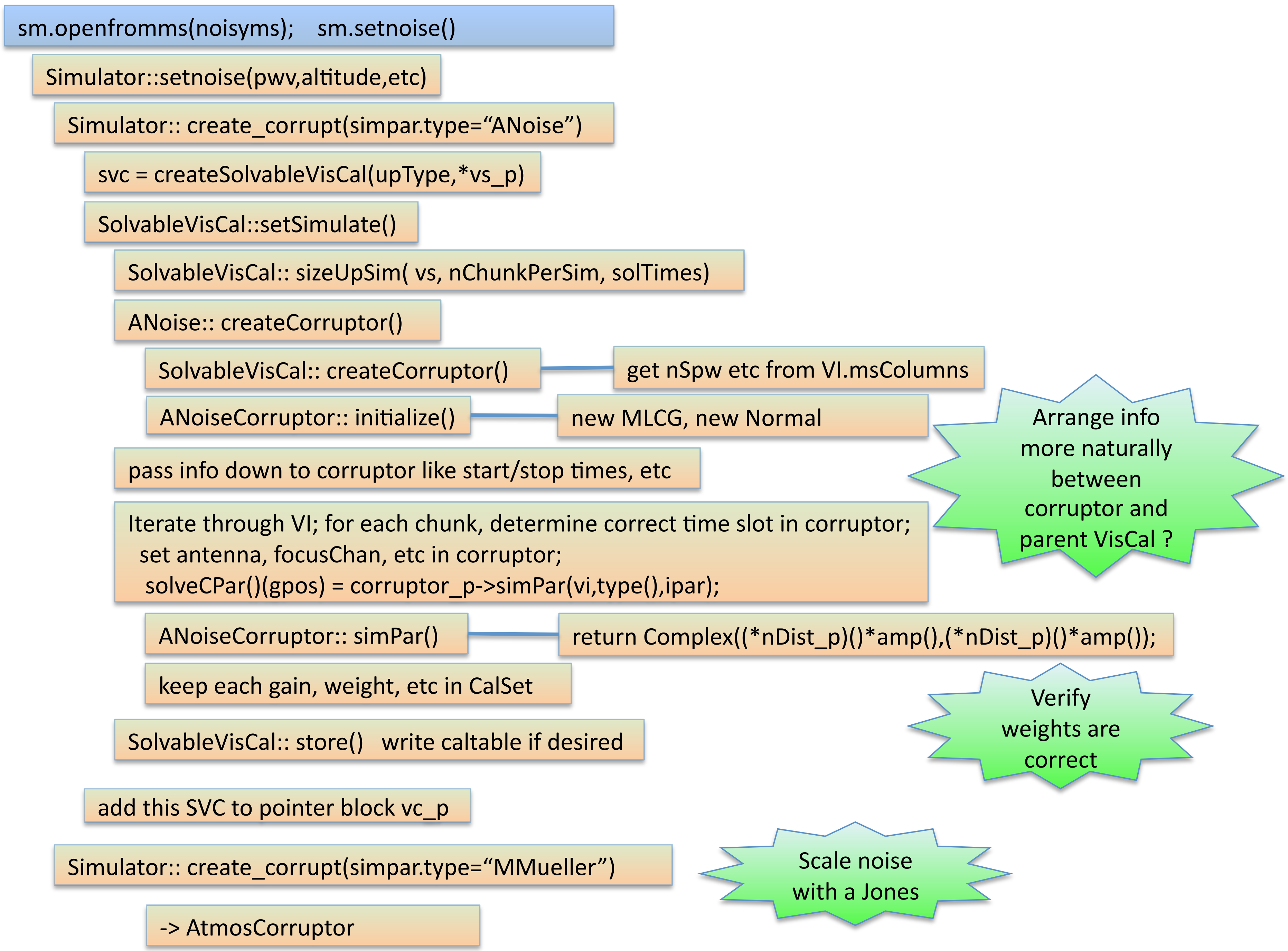
$$= \begin{pmatrix} J^{p\rightarrow p} s^p + J^{q\rightarrow p} s^q \\ J^{p\rightarrow q} s^p + J^{q\rightarrow q} s^q \end{pmatrix}_i$$

# Calibration and Corruption

- $J_i$ contains many components:
  - $F$ = ionospheric effects
  - $T$ = tropospheric effects
  - $P$ = parallactic angle
  - $X$ = linear polarization position angle
  - $E$ = antenna voltage pattern
  - $D$ = polarization leakage
  - $G$ = electronic gain
  - $B$ = bandpass response
  - $K$ = geometric compensation
  - M, A = baseline-based corrections

$$\overleftrightarrow{J}_i = \overleftrightarrow{K}_i \overleftrightarrow{B}_i \overleftrightarrow{G}_i \overleftrightarrow{D}_i \overleftrightarrow{E}_i \overleftrightarrow{X}_i \overleftrightarrow{P}_i \overleftrightarrow{T}_i \overleftrightarrow{F}_i$$

- Order of terms follows signal path (right to left)
- In CASA, each term is a VisCal, and their application to visibilities is handled by the VisEquation
- For simulation, we must create VisCals of the the desired types, calculate their terms a priori and store that information in their CalSets


sm.openfromms(noisyms); sm.setnoise()
Simulator::setnoise(pwv,altitude,etc)
Simulator:: create_corrupt(simpar.type="ANoise")
svc = createSolvableVisCal(upType,*vs_p)
SolvableVisCal::setSimulate()
SolvableVisCal:: sizeUpSim( vs, nChunkPerSim, solTimes)
ANoise:: createCorruptor()
SolvableVisCal:: createCorruptor()
get nSpw etc from VI.msColumns
ANoiseCorruptor:: initialize()
new MLCG, new Normal
Arrange info more naturally between corruptor and parent VisCal ?
pass info down to corruptor like start/stop times, etc
Iterate through VI; for each chunk, determine correct time slot in corruptor;
set antenna, focusChan, etc in corruptor;
solveCPar()(gpos) = corruptor_p->simPar(vi,type(),ipar);
ANoiseCorruptor:: simPar()
return Complex((*nDist_p)()*amp(),(*nDist_p)()*amp());
keep each gain, weight, etc in CalSet
Verify weights are correct
SolvableVisCal:: store() write caltable if desired
add this SVC to pointer block vc_p
Simulator:: create_corrupt(simpar.type="MMueller")
Scale noise with a Jones
-> AtmosCorruptor

- sm.corrupt()
  - Simulator:: corrupt()
    - VisEquation:: setApply(vc_p) puts VCs in order
    - VisEquation:: setPivot() correct Model with some VCs, corrupt Data with the rest
      - Iterate VI
        - VisEquation:: collapseForSim(vb)
          - Model = Data
          - VisCal:: corrupt()
            - e.g. VisMueller:: applyCal(ModelCube)
          - Data = 0
          - VisCal:: correct()
            - e.g. VisMueller:: applyCal(visCube)
          - vb.visCube()+=vb.modelVisCube();
        - VisIter:: setWeightMat(vb) check
    - copy from VB back to VI