Beginner's Guide to Synthesizing Noisy Interferometer Data

Terminology:

- CASA = more precisely casapy, i.e. the CASA binaries with python user interface
- `simdata` *task*: can be run with parameter menu, i.e. `inp simdata`, `go simdata`. The task actually consists of a python script `task_simdata.py` which calls the Simulator tool. You can find that python script in your installation and read precisely what it does.
- Simulator *tool*, accessed from the user interface via `sm`, e.g.

  CASA <> `sm.openfromms(''foobar.ms'')`

  CASA <> `sm.setnoise2(table=''foobar.noise'')`

  CASA <> `sm.corrupt()`

  CASA <> `sm.done()`

  As is typical in CASA, not all *tool* functionality or flexibility is available in the *task*.
- `VisCal` *object*: CASA is object-oriented C++ and python. Each term which can corrupt your data is internally represented by a VisCal, e.g. the BJones VisCal deals with the bandpass shape, both corrupting it, and for real data, correcting for it.

Radiative Transfer:
We'll consider coherent detectors in a minute, i.e. the oscillatory and vector nature of the signal. For now, just consider the specific intensity I_ν, passing through the atmosphere along a path s, with absorption coefficient α_ν cm^{-1} and emissivity j_ν erg cm^{-3} s^{-1} sr^{-1} Hz^{-1}.

$$\frac{dI_\nu}{ds} = -\alpha_\nu I_\nu + j_\nu \tag{1}$$

Making the usual substitutions for optical depth $d\tau_\nu$=$\alpha_\nu$ds and source function S_ν=j_ν/α_ν, and noting that in local thermodynamic equilibrium the source function is the Plank function at the temperature at that point along the path $B_\nu(T_{K,atm}(z))$,

$$\frac{dI_\nu}{d\tau_\nu} = -I_\nu + B_\nu(T_{K,atm}(z)) \tag{2}$$

Next we use the Rayleigh-Jeans expressions

$$I_\nu = \frac{2kT_A(\nu)}{\lambda^2}; B_\nu = \frac{2kT_{atm}}{\lambda^2} \tag{3}$$

$T_A(\nu)$ has little to do with the radiation temperature of the source, which is not restricted to have a thermal spectrum. It is more accurately a *noise* temperature. If one were to place a telescope or feed and detector with small bandwidth $\Delta\nu$ above the atmosphere and measure power P_ν, one could replace that whole system with a resistor at temperature $T_A(\nu)$, and measure the same power. Similarly, the brightness temperature T_{atm} of the atmosphere at the observing frequency ν is not necessarily the actual kinetic temperature of the atmosphere $T_{K,atm}$.

Next we suppress the actual vertical temperature structure of the atmosphere, i.e. we pretend that the source function is constant. CASA uses Juan Pardo's `ATM` library to actually model the atmosphere and calculate the vertically integrated opacity τ_ν and atmospheric brightness temperature

T_{atm}. We do it correctly so you don't have to worry about it. Note that τ_ν is a function of the airmass A=sec(zenith angle). The measured noise temperature T is

$$T(\text{at telescope}) = T_A^*(\text{above atmosphere})e^{-\tau_\nu} + T_{atm}(1 - e^{-\tau_\nu}) \tag{4}$$

Now there's a telescope that is sensitive to the sky with spillover efficiency η_s, and to the surroundings with efficiency $1 - \eta_s$. The surroundings have ambient temperature T_{amb} (somtimes called T_{ground}), so

$$T = \left[T_A^* e^{-\tau_\nu} + T_{atm}(1 - e^{-\tau_\nu})\right] \eta_s + (1 - \eta_s)T_{amb} \tag{5}$$

The cosmic microwave background T_{CMB} is always present (and we make the approximation that it is coming from all directions equally, as is noise from the receiver T_{RX}:

$$\begin{aligned} T &= \left[T_A^* e^{-\tau_\nu} + T_{atm}(1 - e^{-\tau_\nu})\right] \eta_s + (1 - \eta_s)T_{amb} + e^{-\tau_\nu} T_{CMB} + T_{RX} && (6) \\ &= \eta_s T_A^* e^{-\tau_\nu} + T_{CMB} e^{-\tau_\nu} + \eta_s T_{atm}(1 - e^{-\tau_\nu}) + (1 - \eta_s)T_{amb} + T_{RX} && (7) \\ &\equiv T_A + T_S && (8) \end{aligned}$$

Interferometry:

Now we have to consider the cross-correlation of two measured signals T_1 and T_2, which has two components: one proportional to the harmonic mean of the autocorrelations of the two total signals, $T_1 T_2 = (T_{A1} + T_{S1})(T_{A2} + T_{S2})$, i.e. the contribution from each telescope being independently noisy. The second term is proportional to the harmonic mean of the cross-correlations, $T_{A1} T_{A2}$, where only the source signal contributes, since the noise from each antenna does not correlate. (See e.g. Thompson, Moran, and Swenson).

It is frequently assumed that the system noise dominates the source signal (faint sources), so the noise contribution to a measurement is $\sqrt{T_{S1} T_{S2}}$. When expressed in terms of the *source* flux density, *received below the atmosphere*, for integration time Δt and bandwidth $\Delta\nu$, only half of the energy from an unpolarized source is received, so the conversion is

$$F_\nu(\text{noise}) = \frac{2 \times 4k10^{-23}}{\eta_a \eta_c \pi d_1 d_2 \sqrt{2\Delta\nu\Delta t}} \sqrt{T_{S1} T_{S2}} \tag{9}$$

Since the different noise sources don't correlate, we can write

$$\sqrt{T_{S1} T_{S2}} = T_{sys} = T_{CMB} e^{-\tau_\nu} + \eta_s T_{atm}(1 - e^{-\tau_\nu}) + (1 - \eta_s)T_{amb} + T_{RX} \tag{10}$$

where for two antennas with different airmasses A_1 and A_2,

$$\tau_\nu = \tau_{\nu,zenith}(A_1 + A_2)/2 \tag{11}$$

Sometimes it makes sense to express everything in the temperature scale above the atmosphere (T_A^*), or to express noise flux density relative to the source flux density before atmospheric attenuation. This amounts to multiplying the previous expression by $\exp(+\tau_\nu)$:

$$F_\nu^*(\text{noise}) = \frac{4\sqrt{2}k10^{-23}}{\eta_a \eta_c \pi d_1 d_2 \sqrt{\Delta\nu\Delta t}} \left[T_{CMB} + \eta_s T_{atm}(e^{\tau_\nu} - 1) + (1 - \eta_s)T_{amb} e^{\tau_\nu} + T_{RX} e^{\tau_\nu}\right] \tag{12}$$

This is the scale assumed for simulated data - Janskies above the atmosphere.

Coherent detection:
The signals are measured with phase and polarization direction. The atmosphere introduces a phase delay at each antenna ϕ_1, ϕ_2, assumed independent of polarization, so the complex visibility is corrupted by multiplying by $\exp(i(\phi_1 - \phi_2))$.

Implementation in CASA:

CASA (and most synthesis data software) is based on the Measurement Equation method of calibration. Operations are performed on a complex visibility vector (e.g. two complex numbers, one for each polarization). Each potential effect on the data (atmospheric phase noise, receiver gain, etc) is modeled by multiplying the visibility vector by a matrix. (Or in the case of thermal noise, adding a noise vector). So the **B** Jones matrix represents the bandpass shape, the **G** Jones matrix time-dependent complex gain calibration, etc. The calibration of real data involves using redundancy in the data or prior information to solve for the matrices, and then multiplying the real data by the inverse of those matrices. CASA corrupts simulated data by inverting the process. "Perfect" noise-free visibilities are calculated by Fourier-transforming the user's model of the sky, and then those are multiplied by synthesized calibration/corruption matrices. One can, within limitations of the Meaurement Equation formalism (not all matrices commute), and whether we've implemented all of them yet, do some of each - generate perfect synthetic visibilities, *corrupt* them with e.g. G and T terms, then generate noise, *correct* the noise by the residual effects of calibration, e.g. errors in bandpass calibration, and finally add that to the visibilities.

Since CASA is object-oriented, each calibration matrix is represented by a `VisCal` object. Each VisCal is intended to encapsulate a different physical effect on the data. Note that the separation between different effects may be inter-related, and not separable, so that although we claim to be able to separately calibrate gain fluctuations shared by both polarizations from cross-polarization effects, and the calibration machinery heuristically will succeed by doing it this way, real life may be harder. This is important because simulated VisCals will correspond exactly to one physical effect, which may not represent real life.

Each VisCal is stored in a `cal table` - if the user creates one with `simdata` or `Simulator`, they can then use `plotcal` (and in the future `plotms`) to plot it. For example the plot of a `TJones` (see below) would show the gain fluctuations with time as the atmospheric phase delay screen passes over the array

- `TOpac` VisCal = attenuation by the troposphere $\exp(-\tau_\nu)$.
  * Currently not frequency dependent. Need to implement a frequency-dependent `TfOpac` using the `ATM` model. (expected 3.0)
- `TJones`: can introduce atmospheric phase fluctuations either individually for each antenna, or by blowing a 2d screen over the array at a specified windspeed. The user can create this corruption with `sm.settrop`.
  * Although it does use `ATM` to determine the phase as a function of frequency and pwv fluctuation, this should happen as a TfJones VisCal rather than TJones. (expected 3.0)
  * Better interpolation of the delay screen is a future improvement (expected 3.0patch)
  * It would be desirable to attach this to `simdata` (probably 3.0patch)
- `ANoise`: additive random noise. Currently only scaled to $1/\sqrt{\Delta t \Delta \nu}$. The user adds thermal noise with `sm.setnoise2`, or from `simdata`, which creates constant-amplitude noise with

`ANoise`, then scales it according to T_{sys} using an `MfMueller`.
* still a few bugs, to be fixed for 3.0 and the scaling needs to be to be changed to a `TfJones`.

- `DJones`: a constant cross-polarization, user-accessed with `sm.setleakage`.
- `GJones`: fluctuations in complex receiver gain, independent for each telescope, accessible with `sm.setgain`
- `BJones`: ** not yet implemented ** user-specified bandpass shape (probably 3.0patch)
- `EPJones`: ** not yet implemented ** pointing errors - Pointing errors can be calculated as a time-dependent offset or fluctuation in the primary beam pattern. CASA's treatment of pointing errors is a long-term project of Sanjay's.
- antenna-dependent feed position angle: ** not yet implemented ** This can be implemented in CASA by varying the voltage pattern for each antenna. The internal specification of votlage patterns is currently under development by Rob, and implementation of this in Simulator will require consultation by George.

Appendix: sdsim and SimACohCalc:

Single-dish / total power simulation is done in CASA using the `sdsim` task. Since total power data is often stored in a Measurement Set in the autocorrelation rows, and the CASA calibration mechanism currently does not act on those rows, we cannot yet use the same `VisCal`-based corruptions. For thermal nose we use a different class called `SimACohCalc`.

This internal subtlety should cause only minimal differences in noise amplitude compared to `simdata` used with `mode = ``tsys-manual''` (`SimACohCalc` does not interface with the ATM library). We use the same routine `simutil.noisetemp()` to determine the receiver temperature and efficiencies, and the noise amplitude is calculated using the same formula 12, with one difference: the sky and ground temperatures T_{atm} and T_{amb} are assumed equal, so the equation simplifies to:

$$F_\nu^*(\text{noise}) = \frac{4\sqrt{2}k10^{-23}}{\eta_a \eta_c \pi d^2 \sqrt{\Delta\nu\Delta t}} \left[T_{CMB} + \eta_s T_{atm}(e^{\tau_\nu} - \eta_s) + T_{RX} e^{\tau_\nu}\right] \tag{13}$$