ME4: Extended Sources, Images, Image-Plane Effects

Objectives:

- Learning to predict extended sources
- Getting to grips with wide fields and *n*
- Thinking about image-plane effects
- *Planting the seeds* of some advanced topics and techniques

svn up Workshop2007 please

So, Is There Always Such A Beast As "Corrected" *uv*-Data?

Say we now have some image-plane effects:

$$\mathbf{V}_{pq} = \mathbf{G}_p \mathcal{F}(N_p \mathbf{E}_p \mathbf{B} \mathbf{E}_q^\dagger N_q^\dagger) \mathbf{G}_q^\dagger$$

... and we know all of the $\mathbf{G}_p, \mathbf{E}_p$, and (of course) N_p's;
then is there a way to obtain "corrected" visibilities $\mathbf{V}'$

such that $\mathbf{V}'_{pq} = \mathcal{F}(\mathbf{B})$???

(or at the very least $\mathbf{V}'_{pq} = \mathcal{F}(N_p \mathbf{B} N_q^\dagger)$) ???

And The Answer Is...

- In general, **NO!**
- *uv*-plane effects (the ***G***s) can be taken out.
- Image-plane effects correspond to convolution in the *uv*-plane:

$$\mathbf{V}_{pq} = \mathcal{F}(N_p \mathbf{E}_p \mathbf{B} \mathbf{E}_q^\dagger N_q^\dagger) = \mathcal{F}(\mathbf{E}_p) \circ \mathcal{F}(N_p \mathbf{B} N_q^\dagger) \circ \mathcal{F}(\mathbf{E}_q^\dagger)$$

  - ...with time-variable kernels
  - ...and with each baseline's *uv*-plane sampled along just a single track

(Note: Bhatnagar et al. (EVLA Memo 100) suggest a method for <u>approximate</u> correction during the imaging step. We'll return to this later.)

Correcting At A Single Point

But we can correct for a single point l_0, m_0:

$$\mathbf{V}' = \mathbf{E}_p(l_0, m_0)^{-1} \mathbf{V} \mathbf{E}_q^{-1}(l_0, m_0)^\dagger =$$

$$= \mathcal{F}\left((\mathbf{E}_p(l_0, m_0))^{-1} \mathbf{E}_p N_p \mathbf{B} N_q^\dagger \mathbf{E}_q^\dagger (\mathbf{E}_q^\dagger(l_0, m_0))^{-1} \right)$$

$$= \mathcal{F}(N_p \mathbf{B}' N_q^\dagger)$$

where $\mathbf{B}'(l_0, m_0) = \mathbf{B}(l_0, m_0)$, but diverges further away.

- In general, *uv*-data can only be "corrected" for a single point on the sky.
- This is the motivation for facet imaging.

Divide And Conquer

Let's forget about the $\boldsymbol{G}$s, as they're trivial to put in/remove.

(M.E.) $$\boldsymbol{V}_{pq}=\iint_{lm}\boldsymbol{K}_p\boldsymbol{E}_p\frac{\boldsymbol{B}}{n}\boldsymbol{E}_q^\dagger\boldsymbol{K}_q^\dagger\,dl\,dm=\int_{\text{sky}}\boldsymbol{K}_p\boldsymbol{E}_p\boldsymbol{B}\boldsymbol{E}_q^\dagger\boldsymbol{K}_q^\dagger\,d\Omega$$

This is linear over $\boldsymbol{B}$, so if the sky is a sum of sources:

$\boldsymbol{B}(l,m)=\sum_s\boldsymbol{B}_s(l,m)$, (or $\boldsymbol{B}(\vec{\sigma})=\sum_s\boldsymbol{B}_s(\vec{\sigma})$)

then

$$\boldsymbol{V}_{pq}=\sum_s\boldsymbol{V}_{pq}^{(s)}=\sum_s\iint_{lm}\boldsymbol{K}_p\boldsymbol{E}_p\frac{\boldsymbol{B}_s}{n}\boldsymbol{E}_q^\dagger\boldsymbol{K}_q^\dagger\,dl\,dm$$

And we can examine the integral for each source separately.

Case 1. Point Sources

- Point sources we have already done, their $\boldsymbol{B}$ is a delta function.

$$\boldsymbol{V}_{pq}=\iint_{lm}\boldsymbol{K}_p\boldsymbol{E}_p\boldsymbol{B}_0\,\delta(l-l_0,m-m_0)\boldsymbol{E}_q^\dagger\boldsymbol{K}_q^\dagger\,dl\,dm\;=$$

$$=\;\boldsymbol{K}_p(l_0,m_0)\boldsymbol{E}_p(l_0,m_0)\boldsymbol{B}_0\boldsymbol{E}_q^\dagger(l_0,m_0)\boldsymbol{K}_q^\dagger(l_0,m_0)\;=$$

$$=\;\boldsymbol{K}_{p0}\boldsymbol{E}_{p0}\boldsymbol{B}_0\boldsymbol{E}_{q0}^\dagger\boldsymbol{K}_{q0}^\dagger$$

- Product of five 2x2 matrices.
- We can predict point sources perfectly.
- Computationally practical for a limited number of sources.

Case 2. Extended Sources And Patches

- Any $\boldsymbol{B}$ that is not a delta-function gets interesting, especially if $\boldsymbol{E}$ is not trivial, and especially for wider fields (the “not quite an F.T.” problem.)
- ...but we need to solve this for two reasons.
- Reason 1: spatially extended sources...
- Reason 2: it is impractical to directly predict large numbers of fainter sources individually, so we need to organize them into “patches”, and find a faster way to predict a patch *en masse*.

First, Reduce To Center...

Consider a source centered on direction $\vec{\sigma}_0$ (l_0,m_0),

with the phase center at $\vec{\sigma}=0$.

$$\boldsymbol{V}_{pq}=\int_{\text{sky}}\boldsymbol{K}_p(\vec{\sigma})\boldsymbol{E}_p(\vec{\sigma})\boldsymbol{B}(\vec{\sigma})\boldsymbol{E}_q^\dagger(\vec{\sigma})\boldsymbol{K}_q^\dagger(\vec{\sigma})\,d\Omega=$$

Let's integrate in the coordinate system $\vec{\sigma}'=\vec{\sigma}+\vec{\sigma}_0$,

and define $\tilde{\boldsymbol{B}}(\vec{\sigma})\equiv\boldsymbol{B}(\vec{\sigma}+\vec{\sigma}_0)$, $\tilde{\boldsymbol{E}}_p(\vec{\sigma})\equiv\boldsymbol{B}(\vec{\sigma}+\vec{\sigma}_0)$:

$$\boldsymbol{V}_{pq}=\int_{\text{sky}}\boldsymbol{K}_p(\vec{\sigma}+\vec{\sigma}_0)\boldsymbol{E}_p(\vec{\sigma}+\vec{\sigma}_0)\boldsymbol{B}(\vec{\sigma}+\vec{\sigma}_0)\boldsymbol{E}_q^\dagger(\vec{\sigma}+\vec{\sigma}_0)\boldsymbol{K}_q^\dagger(\vec{\sigma}+\vec{\sigma}_0)\,d\Omega'=$$

$$=\boldsymbol{K}_p(\vec{\sigma}_0)\left(\int_{\text{sky}}\boldsymbol{K}_p(\vec{\sigma})\tilde{\boldsymbol{E}}_p(\vec{\sigma})\tilde{\boldsymbol{B}}(\vec{\sigma})\tilde{\boldsymbol{E}}_q^\dagger(\vec{\sigma})\boldsymbol{K}_q^\dagger(\vec{\sigma})\,d\Omega\right)\boldsymbol{K}_q^\dagger(\vec{\sigma}_0)=$$

$$=\boldsymbol{K}_p(l_0,m_0)\left(\iint_{lm}\boldsymbol{K}_p\tilde{\boldsymbol{E}}_p\frac{\tilde{\boldsymbol{B}}}{n}\tilde{\boldsymbol{E}}_q^\dagger\boldsymbol{K}_q^\dagger\,dl\,dm\right)\boldsymbol{K}_q^\dagger(l_0,m_0)$$

...Where Life Gets Easier

$$V_{pq} = K_{p0} \left(\iint_{lm} K_p \tilde{E}_p \frac{\tilde{B}}{n} \tilde{E}_q^\dagger K_q^\dagger \, dl \, dm \right) K_{q0}^\dagger$$

- In other words, to predict a source at l_0, m_0, it is sufficient to predict it at the phase center (shifting $\boldsymbol{E}$ accordingly...), then apply the phase terms $\boldsymbol{K}_{p0}$, $\boldsymbol{K}_{q0}$ to "move" the source to its true position.
- This is beautifully symmetric w.r.t. the ME for a point source:

$$K_{p0} E_{p0} B_0 E_{q0}^\dagger K_{q0}^\dagger$$

So All We Need To Figure Out Now...

- ...is how to predict extended sources at the phase center. We've simplified the problem considerably. If we now expand the $\boldsymbol{K}$ terms:

$$V_{pq} = \iint_{lm} \left(E_p \frac{B}{n} E_q^\dagger \right) e^{-2\pi i (u_{pq} l + v_{pq} m + w_{pq}(n-1))} \, dl \, dm$$

When $n \to 1$, this becomes a Fourier Transform:

$$V_{pq} = \iint_{lm} (\underbrace{E_p B E_q^\dagger}_{\text{"apparent sky"}}) \underbrace{e^{-2\pi i (u_{pq} l + v_{pq} m)}}_{\text{F.T. kernel}} \, dl \, dm$$

- ...and we only require the "Weak Assumption" that $n \to 1$ <u>over the extent of the source.</u>

Or Even More Optimistically...

If we also make the "Strong Assumption" that $E_p \approx E_p(0,0)$ over the source, then

$$V_{pq} = K_{p0} E_{p0} \mathcal{F}(B_0) E_{q0}^\dagger K_{q0}^\dagger$$

- This is very good from a practical point of view, since:
  - F.T.'s are easier than strange integrals
  - phase shifts and (in the optimistic case) image-plane effects are applied in the same way to all kinds of sources
  - we only need to worry about predicting the source (or patch) at the phase center

The Bad News

- This is only an approximation.
- Error levels will depend on:
  - source flux (in a linear way, so we can talk about relative error);
  - patch/source size (because of the *n* term);
  - the variation in $\boldsymbol{E}$ over the extent of the patch.
- We need to develop some feeling for how small the patches/sources need to be to make the error suitably small.
- We can build some trees to simulate these errors (well this is actually good news, isn't it?)

Case 2: "Analytic" Extended Sources

- For some kinds of sources (e.g., a 2D Gaussian), we can write out the F.T. analytically.

i.e. given a set of source parameters that determine its $\boldsymbol{B}$ (for a Gaussian, this is its major/minor axis extent $\sigma_{1,}\sigma_2$, position angle θ, and integrated I,Q,U,V fluxes), we can immediately write out an expression for its $\mathcal{F}(\boldsymbol{B})$. (The F.T. of a Gaussian is a Gaussian.)

- Meow.GaussianSource implements the trees for this.
- Only correct under the "Strong Assumption", obviously.

Shapelets: The Ultimate In Analytic Sources

- The shapelet transform decomposes an image (in *l*,*m*) into a sum of *shapelet basis functions:*

$$b(l,m)=\sum_n c_n \cdot s_n(l,m)$$

- Complex source structure can (potentially) be represented with relatively few low-order $\{c_n\}$ coefficients.
- The basis functions s_n have an analytic F.T.
- Therefore, we can quickly compute the F.T. of a shapelet source, given a set of coefficients.
- "Strong Assumption" not necessarily required...
- Just making its way into MeqTrees, so no demo...

Case 3: "Image" Sources

- Finally, we can have an image – $\boldsymbol{B}$ (or perhaps just plain *I*) sampled on a RA/Dec (or other) grid.
- An FFT gives us the F.T. on a regular *uv* grid:

$$\{\boldsymbol{B}_{ij}=\boldsymbol{B}(l_i,m_j)\} \overset{\text{FFT}}{\rightarrow} \{\boldsymbol{X}_{ij}=\mathcal{F}(\boldsymbol{B})(u_i,v_j)\}$$

- This is done by the **Meq.FFTBrick** node.
- The **Meq.UVInterpol** node then takes...

...$u(t), v(t)$ from one child, and interpolates $\boldsymbol{X}(t,\nu)=\boldsymbol{X}\left(\frac{\nu u(t)}{c},\frac{\nu v(t)}{c}\right)$ within the $\{\boldsymbol{X}_{ij}\}$ "brick" (which it gets from the other child.)

Implementing An Image Source

- **Meow.FITSImageComponent** implements the right combination of **Meq.FFTBrick**, **Meq.UVInterpol,** and **Meq.FITSImage**.
- Images can also have a frequency axis, this is seamlessly propagated through the tree.
- No demo – something is currently broken :(

Bonus Image: The True nJy Sky

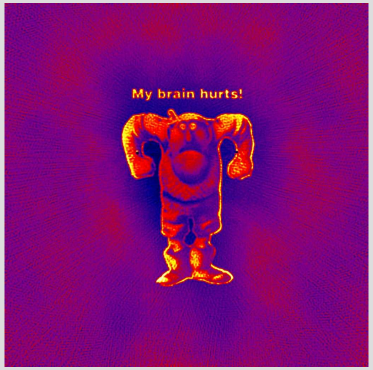

Image Sources & Image-Plane Effects

Given an image, $\{\boldsymbol{B}_{ij}\}$, we can easily apply $\boldsymbol{E}$:

$$\boldsymbol{B}_{pq}=\boldsymbol{E}_p \boldsymbol{B} \boldsymbol{E}_q^{\dagger}$$

- This can be done in a simple tree:
  - the **FFTBrick** node expects a “brick” child.
  - this doesn't have to be an actual **FITSImage** node, but can also be any subtree, i.e. one that does some operations on the image.
- If the $\boldsymbol{E}_p$'s are different per antenna, this is INSANELY expensive (an FFT per baseline!)

The Bhatnagar Approach:

Coming Soon To a Tree Near You?

$$\mathcal{F}(\boldsymbol{E}_p \boldsymbol{B} \boldsymbol{E}_q^{\dagger})=\mathcal{F}(\boldsymbol{E}_p)\circ \boldsymbol{X} \circ \mathcal{F}(\boldsymbol{E}_q^{\dagger})$$

If $\boldsymbol{E}_p$ is sufficiently smooth, then $\mathcal{F}(\boldsymbol{E}_p)$ can be approximated with just a few low-order Fourier coefficients.

- Convolution with a “small” kernel can be computed directly, fairly cheaply.
- ...and in fact the interpolation function used inside **UVInterpol** to interpolate $\boldsymbol{X}$ to the *uv* track is a form of convolution.
- So we can apply $\boldsymbol{E}$'s with a suitable modification to the convolution function.

n: Naughty, Nasty, Notorious, Nefarious, Nebulous, Noxious, or Just Negligible?

- The “Weak Assumption” is that n=1 over the extent of the source.
- Let's make a differential tree to estimate the error that this assumption introduces.
- General idea: make a test grid of point sources, then:
  - predict them perfectly in one branch.
  - predict them with (*l,m,1*) in the other branch.
- The former are Meow.PointSources, how do we implement the latter?
- ...through OOP, of course!

A Naughty Direction

- *lmn*'s ultimately come from Meow.LMDirection.
- We're going to *derive* a subclass from Meow.LMDirection
- This subclass *reimplements* the **n()** function, which LMDirection uses to create its *n* node (which is subsequently used to make a ***K*** term).
- The end result: a direction that “ignores” the normal *n* term, and uses 1 instead.
- See `ME4/NaughtyDirection.py`.

Demo 1: So How Weak Is The “Weak Assumption”?

- Load up `ME4/demo1-naughty.py`.
- This makes two sets of sources arranged in a “cross”:
  - Regular PointSources in one branch
  - PointSources with a NaughtyDirection in the other branch
- You can select the grid stepping size – compare results for 1' and 60'.
- Compare MFS and per-channel maps.

Mega-Exercise 3: Replicating The Perley Movie

The ionosphere introduces two effects: Faraday rotation $\boldsymbol{F}=\mathrm{Rot}(\frac{\mathrm{RM}}{\nu^2})$, and phase delay: $\boldsymbol{L}=e^{-i\theta}$.

Both RM and θ are proportional to TEC(l,m).

Let's simulate a TID (Travelling Ionospheric Disturbance), by making $TEC(x,y)$ constant + a moving 1D sine wave. (And each antenna looks up through a different x,y point.)

We'll make a grid of point sources, and apply $\boldsymbol{F}$ and $\boldsymbol{L}$.

With a bit of luck, we can even force the imager to make a series of time slices...