CASA VLBI

WORKSHOP 2020

2-6 NOVEMBER 2020

LECTURE 11: POLARIZATION CALIBRATION

I. MARTI-VIDAL

(University of Valencia, SPAIN)

CASA Common Astronomy Software Applications

JIVE Joint Institute for VLBI ERIC

RadioNet

JUMPING JIVE Joint Institute for VLBI ERIC

Polarization Calibration in VLBI

AIPS tools and status of CASA tools

Ivan Martí-Vidal

Observatori Astronòmic & Dpt. Astronomia i Astrofísica
Universitat de València
(GIDEGENT Research Program, GVA)

CASA-VLBI Workshop 2020 — JIVE (Netherlands)

VNIVERSITAT DE VALÈNCIA

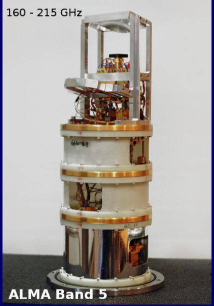
JIVE

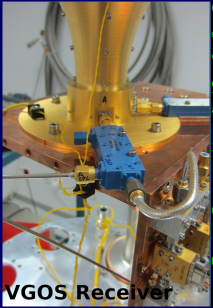
JUMPING JIVE
Joint Institute for VLBI ERIC

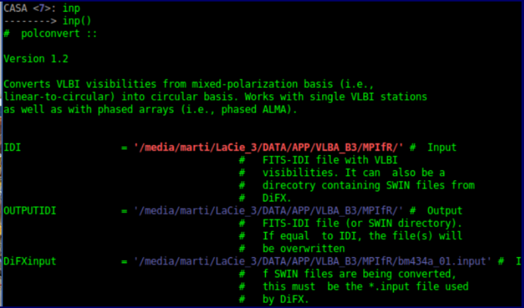
RadioNet

Basic Concepts

Dterms: The Root of All Evil







- The signal is split into orthogonal polarizations (e.g., OMTs, reflection gratings, T-septums, ...)
- Polarizations can also be converted in different ways (e.g., quarter waveplates, software).
- None of these devices is perfect. Signals from one polarization are leaked into the other.
- Such a leakage (with a given amplitude and delay/phase) is modelled with the Dterms.

The MEq: A Full-Stokes Interferometry Formalism

For a source with a generic (polarized) brightness distribution, the visibility matrix between two antennas a and b (with no DDEs) is

$$\mathcal{V}_{ab}^{obs} = J_a \left[\int_{\alpha,\delta} \mathcal{S}\, e^{-\frac{2\pi j}{\lambda}(u\,\alpha + v\,\delta)} \frac{d\alpha\, d\delta}{z} \right] (J_b)^H ,$$

where J are the calibration Jones matrices (e.g., Smirnov 2011).

The MEq: A Full-Stokes Interferometry Formalism

For a source with a generic (polarized) brightness distribution, the visibility matrix between two antennas a and b (with no DDEs) is

$$\mathcal{V}_{ab}^{obs} = J_a \left[\int_{\alpha,\delta} \mathcal{S}\, e^{-\frac{2\pi j}{\lambda}(u\,\alpha + v\,\delta)} \frac{d\alpha\, d\delta}{z} \right] (J_b)^H ,$$

where J are the calibration Jones matrices (e.g., Smirnov 2011).

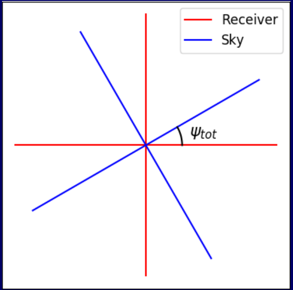
Let us remember the classical interferometer equation:

$$V_{ab}^{obs} = G_a\, G_b^* \int_{\alpha,\delta} I(\alpha,\delta)\, e^{-\frac{2\pi j}{\lambda}(u\,\alpha + v\,\delta)} \frac{d\alpha\, d\delta}{z}$$

Receiver Frame vs. Sky Frame

$$P_{xy} = \begin{pmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{pmatrix} \qquad P_{rl} = \begin{pmatrix} e^{j\psi} & 0 \\ 0 & e^{-j\psi} \end{pmatrix}$$

- The parallactic angle, ψ, is the (time-dependent) rotation of the antenna-mount axis w.r.t. the sky.
- It is deterministic. In VLBI, it's good to apply it before the phase (and delay/rate) calibration.
- The Receiver and Sky frames are related by a rotation $\psi_{tot} = \psi + \psi_0$ (where ψ_0 is the antenna feed angle).



Antenna Feeds with Circular Polarizers

The visibility matrix (Rec. Frame) is

$$\mathcal{V}_{ab} = \begin{pmatrix} R_a R_b^* & R_a L_b^* \\ L_a R_b^* & L_a L_b^* \end{pmatrix}$$

The brightness matrix (Sky Frame) is

$$\mathcal{S} = \begin{pmatrix} I+V & Q+jU \\ Q-jU & I-V \end{pmatrix}$$

Antenna Feeds with Circular Polarizers

The visibility matrix (Rec. Frame) is

$$\mathcal{V}_{ab} = \begin{pmatrix} R_a R_b^* & R_a L_b^* \\ L_a R_b^* & L_a L_b^* \end{pmatrix}$$

The brightness matrix (Sky Frame) is

$$\mathcal{S} = \begin{pmatrix} I+V & Q+jU \\ Q-jU & I-V \end{pmatrix}$$

Receiver Frame and Sky Frame are related by:

Rotation matrix: $P(\psi_{tot}) = \begin{pmatrix} e^{j\psi_{tot}} & 0 \\ 0 & e^{-j\psi_{tot}} \end{pmatrix}$

Antenna Feeds with Circular Polarizers

The visibility matrix (Rec. Frame) is

$$\mathcal{V}_{ab} = \begin{pmatrix} R_a R_b^* & R_a L_b^* \\ L_a R_b^* & L_a L_b^* \end{pmatrix}$$

The brightness matrix (Sky Frame) is

$$\mathcal{S} = \begin{pmatrix} I+V & Q+jU \\ Q-jU & I-V \end{pmatrix}$$

Receiver Frame and Sky Frame are related by:

Rotation matrix: $P(\psi_{tot}) = \begin{pmatrix} e^{j\psi_{tot}} & 0 \\ 0 & e^{-j\psi_{tot}} \end{pmatrix}$

$$\mathcal{V}_{ab}^{Sky} = P(-\psi_{tot}^a)\mathcal{V}_{ab}^{Rec} P(\psi_{tot}^b)$$

$$\mathcal{S}_{ab}^{Rec} = P(\psi_{tot}^a)\mathcal{S}_{ab}^{Sky} P(-\psi_{tot}^b) = \begin{pmatrix} (I+V)\,e^{j\delta} & (Q+jU)\,e^{j\Delta} \\ (Q-jU)\,e^{-j\Delta} & (I-V)\,e^{-j\delta} \end{pmatrix}$$

where $\Delta = \psi_{tot}^a + \psi_{tot}^b$ and $\delta = \psi_{tot}^a - \psi_{tot}^b$

Jones Calibration Matrices. Examples

- Gain, $G^{Rec} = \begin{pmatrix} A_R(t)\, e^{j\phi_x(t)} & 0 \\ 0 & A_L(t)\, e^{j\phi_y(t)} \end{pmatrix}$

- Bandpass, $B^{Rec} = \begin{pmatrix} A_R(\nu)\, e^{j\phi_x(\nu)} & 0 \\ 0 & A_L(\nu)\, e^{j\phi_y(\nu)} \end{pmatrix}$

- Pol. leakage $D^{Rec} = \begin{pmatrix} 1 & D_R(\nu) \\ D_L(\nu) & 1 \end{pmatrix}$

Jones Calibration Matrices. Examples

- Gain, $G^{Rec} = \begin{pmatrix} A_R(t)\, e^{j\phi_x(t)} & 0 \\ 0 & A_L(t)\, e^{j\phi_y(t)} \end{pmatrix}$
- Bandpass, $B^{Rec} = \begin{pmatrix} A_R(\nu)\, e^{j\phi_x(\nu)} & 0 \\ 0 & A_L(\nu)\, e^{j\phi_y(\nu)} \end{pmatrix}$
- Pol. leakage $D^{Rec} = \begin{pmatrix} 1 & D_R(\nu) \\ D_L(\nu) & 1 \end{pmatrix}$
- NOTE 1: The Jones matrices are defined in (and should be applied on) the **Receiver Frame**.

Jones Calibration Matrices. Examples

- Gain, $G^{Rec} = \begin{pmatrix} A_R(t)\, e^{j\phi_x(t)} & 0 \\ 0 & A_L(t)\, e^{j\phi_y(t)} \end{pmatrix}$

- Bandpass, $B^{Rec} = \begin{pmatrix} A_R(\nu)\, e^{j\phi_x(\nu)} & 0 \\ 0 & A_L(\nu)\, e^{j\phi_y(\nu)} \end{pmatrix}$

- Pol. leakage $D^{Rec} = \begin{pmatrix} 1 & D_R(\nu) \\ D_L(\nu) & 1 \end{pmatrix}$

- NOTE 1: The Jones matrices are defined in (and should be applied on) the **Receiver Frame**.
- NOTE 2: For circular feeds, the parallactic-angle correction is a diagonal matrix, which conmutes with G^{Rec} and B^{Rec}. Hence, we do not care much about the difference between receiver and sky frames.

Changing Frame on Jones Matrices

$$\mathcal{V}_{cal}^{Sky} = J^{Sky}\,\mathcal{V}_{uncal}^{Sky}$$

Changing Frame on Jones Matrices

$$\mathcal{V}_{cal}^{Sky} = J^{Sky}\,\mathcal{V}_{uncal}^{Sky} \rightarrow \quad \mathcal{V}_{cal}^{Sky} = P(\psi_{tot})\,J^{Rec}\,P(-\psi_{tot})\,V_{uncal}^{Sky}$$

Changing Frame on Jones Matrices

$$\mathcal{V}^{Sky}_{cal} = J^{Sky}\,\mathcal{V}^{Sky}_{uncal} \rightarrow \quad \mathcal{V}^{Sky}_{cal} = P(\psi_{tot})\,J^{Rec}\,P(-\psi_{tot})\,V^{Sky}_{uncal}$$

$$J^{Sky}_{a} = P(\psi_{tot})\,J^{Rec}_{a}\,P(-\psi_{tot})$$

Changing Frame on Jones Matrices

$$\mathcal{V}_{cal}^{Sky} = J^{Sky}\,\mathcal{V}_{uncal}^{Sky} \rightarrow \;\; \mathcal{V}_{cal}^{Sky} = P(\psi_{tot})\,J^{Rec}\,P(-\psi_{tot})\,V_{uncal}^{Sky}$$

$$J_a^{Sky} = P(\psi_{tot})\,J_a^{Rec}\,P(-\psi_{tot})$$

- Gain, $G^{Sky} = \begin{pmatrix} A_R(t)\,e^{j\phi_x(t)} & 0 \\ 0 & A_L(t)\,e^{j\phi_y(t)} \end{pmatrix}$
- Pol. leakage $D^{Sky} = \begin{pmatrix} 1 & D_R(\nu)e^{j2\psi_{tot}} \\ D_L(\nu)e^{-j2\psi_{tot}} & 1 \end{pmatrix}$

Changing Frame on Jones Matrices

$$\mathcal{V}^{Sky}_{cal} = J^{Sky}\,\mathcal{V}^{Sky}_{uncal} \rightarrow \mathcal{V}^{Sky}_{cal} = P(\psi_{tot})\,J^{Rec}\,P(-\psi_{tot})\,V^{Sky}_{uncal}$$

$$J^{Sky}_{a} = P(\psi_{tot})\,J^{Rec}_{a}\,P(-\psi_{tot})$$

- Gain, $G^{Sky} = \begin{pmatrix} A_R(t)\,e^{j\phi_x(t)} & 0 \\ 0 & A_L(t)\,e^{j\phi_y(t)} \end{pmatrix}$ $\rightarrow$ Is independent of ψ_{tot}
- Pol. leakage $D^{Sky} = \begin{pmatrix} 1 & D_R(\nu)e^{j2\psi_{tot}} \\ D_L(\nu)e^{-j2\psi_{tot}} & 1 \end{pmatrix}$ $\rightarrow$ Depends on ψ_{tot}!

Changing Frame on Jones Matrices

$$\mathcal{V}_{cal}^{Sky} = J^{Sky}\, \mathcal{V}_{uncal}^{Sky} \rightarrow \;\; \mathcal{V}_{cal}^{Sky} = P(\psi_{tot})\, J^{Rec}\, P(-\psi_{tot})\, V_{uncal}^{Sky}$$

$$J_a^{Sky} = P(\psi_{tot})\, J_a^{Rec}\, P(-\psi_{tot})$$

- Gain, $G^{Sky} = \begin{pmatrix} A_R(t)\, e^{j\phi_x(t)} & 0 \\ 0 & A_L(t)\, e^{j\phi_y(t)} \end{pmatrix} \quad \rightarrow \quad$ Is independent of ψ_{tot}
- Pol. leakage $D^{Sky} = \begin{pmatrix} 1 & D_R(\nu) e^{j2\psi_{tot}} \\ D_L(\nu) e^{-j2\psi_{tot}} & 1 \end{pmatrix} \quad \rightarrow \quad$ Depends on ψ_{tot}!

We can use the time dependence of ψ_{tot} to decouple the effects of source polarization (which are constant on the sky frame) from the instrumental polarization (which is constant on the receiver frame).

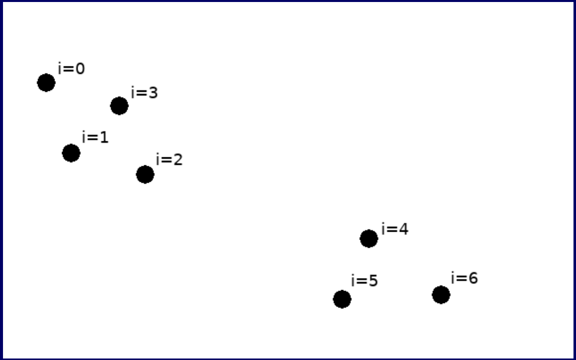
LPCAL and PolSolve: Algorithms and Conventions

VLBI Polarimetry Software

The problem of using spatially-resolved polarization calibrators

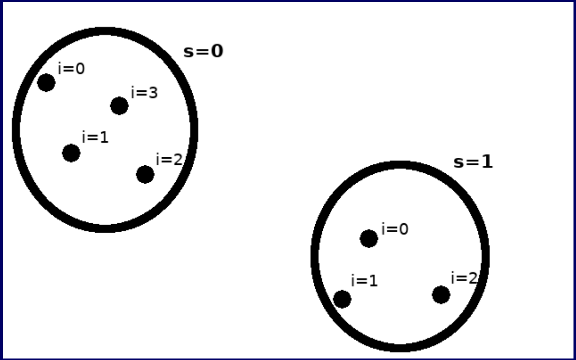
- **Inverse Modelling**.
  - **LPCAL**: Based on AIPS (Leppänen et al. 1995). Pretty old, but well established and tested.
  - **GPCAL**: Based on AIPS/ParselTongue (Park et al. 2020). Overcomes some LPCAL limitations.
  - **PolSolve**: Based on CASA (I. Marti-Vidal et al. 2020). Overcomes some LPCAL limitations.
- **Forward Modelling**.
  - **EHTim** (A. Chael et al. 2018, 2020)
- **MCMC**.
  - **DMC** (D. Pesce 2020) and **THEMIS** (Broderick et al. 2020)

Modelling Polarization Structures I: Slicing



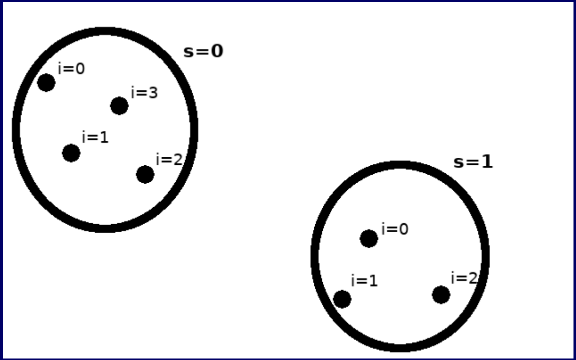
$$\mathcal{V}_I(u,v) = \sum_i^N I_i e^{\frac{2\pi}{\lambda} j(u\alpha_i + v\delta_i)}$$

Modelling Polarization Structures I: Slicing



$$\mathcal{V}_I(u,v) = \sum_s \sum_i^{N_s} I_i^s \, e^{\frac{2\pi}{\lambda} j (u\alpha_i^s + v\delta_i^s)}$$

Modelling Polarization Structures I: Slicing



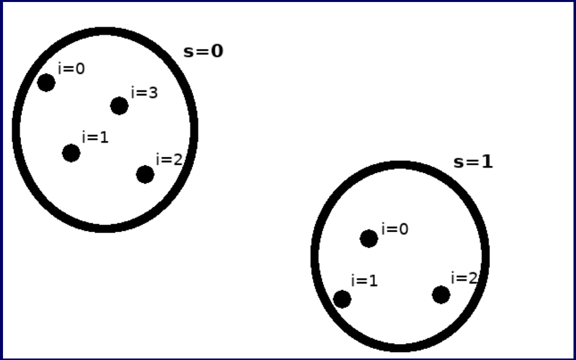
$$\mathcal{V}_I(u,v) = \sum_s \sum_i^{N_s} I_i^s e^{\frac{2\pi}{\lambda} j (u\alpha_i^s + v\delta_i^s)}$$

$$\mathcal{V}_Q(u,v) = \sum_s q_s \left[\sum_i I_i^s e^{\frac{2\pi}{\lambda} j (u\alpha_i^s + v\delta_i^s)} \right]$$

$$\mathcal{V}_U(u,v) = \sum_s u_s \left[\sum_i I_i^s e^{\frac{2\pi}{\lambda} j (u\alpha_i^s + v\delta_i^s)} \right]$$

- We split the set of CLEAN components into disjoint pieces (a.k.a. *subcomponents*) of assumed constant fractional polarization.

Modelling Polarization Structures I: Slicing



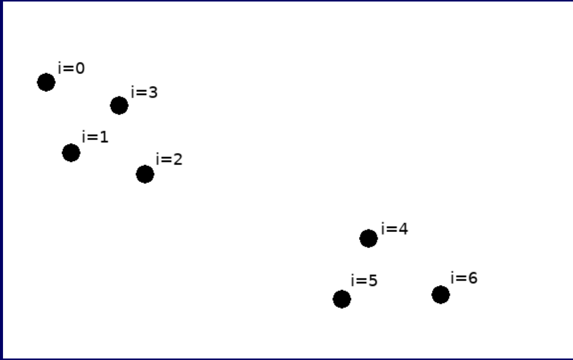
$$\mathcal{V}_I(u,v) = \sum_s \sum_i^{N_s} I_i^s e^{\frac{2\pi}{\lambda} j(u\alpha_i^s + v\delta_i^s)}$$

$$\mathcal{V}_Q(u,v) = \sum_s q_s \left[\sum_i I_i^s e^{\frac{2\pi}{\lambda} j(u\alpha_i^s + v\delta_i^s)} \right]$$

$$\mathcal{V}_U(u,v) = \sum_s u_s \left[\sum_i I_i^s e^{\frac{2\pi}{\lambda} j(u\alpha_i^s + v\delta_i^s)} \right]$$

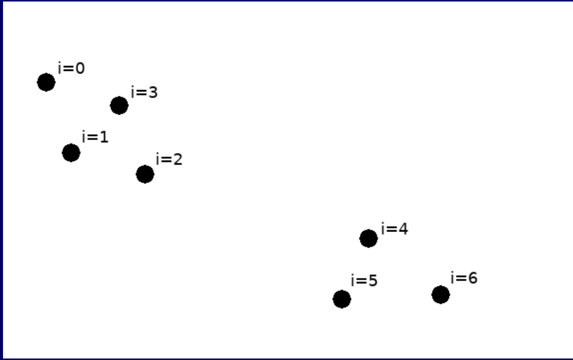
- We split the set of CLEAN components into disjoint pieces (a.k.a. *subcomponents*) of assumed constant fractional polarization.
- We fit the antenna Dterms **and** the subcomponent polarization (q_s and u_s) **together**.

Modelling Polarization Structures I: PolSelfCal



$$\mathcal{V}_I(u,v) = \sum_i^N I_i e^{\frac{2\pi}{\lambda} j (u\alpha_i + v\delta_i)}$$

Modelling Polarization Structures I: PolSelfCal

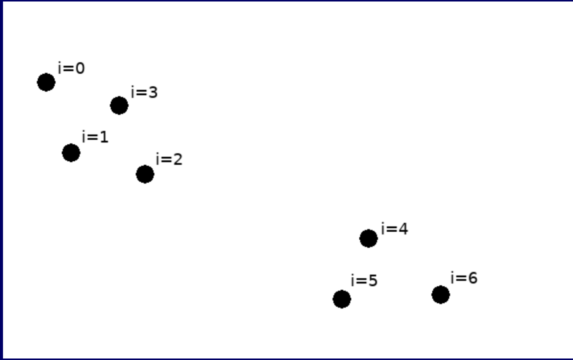


$$\mathcal{V}_I(u,v) = \sum_i^N I_i e^{\frac{2\pi}{\lambda}j(u\alpha_i + v\delta_i)}$$

$$\mathcal{V}_Q(u,v) = \sum_i^N Q_i e^{\frac{2\pi}{\lambda}j(u\alpha_i + v\delta_i)}$$

$$\mathcal{V}_U(u,v) = \sum_i^N U_i e^{\frac{2\pi}{\lambda}j(u\alpha_i + v\delta_i)}$$

Modelling Polarization Structures I: PolSelfCal



$$\mathcal{V}_I(u,v) = \sum_i^N I_i e^{\frac{2\pi}{\lambda} j(u\alpha_i + v\delta_i)}$$

$$\mathcal{V}_Q(u,v) = \sum_i^N Q_i e^{\frac{2\pi}{\lambda} j(u\alpha_i + v\delta_i)}$$

$$\mathcal{V}_U(u,v) = \sum_i^N U_i e^{\frac{2\pi}{\lambda} j(u\alpha_i + v\delta_i)}$$

- We only fit the antenna Dterms, keeping the full-polarization image model fixed. We iterate the procedure (similar to hybrid imaging of Stokes I).

The LPCAL Algorithm

- Fits the Dterms (and source polarization) using a linear approximation (i.e., it does *not* apply the full Meq) and the slicing approach.
- Computes the error function *in the Sky frame*.

$$\chi^2 = \sum_{i,pol} W_i \left(\mathcal{V}^{Sky}_{mod,pol} - \mathcal{V}^{Sky}_{obs,pol} \right)^2_i$$

where *pol* can be *RL* and *LR*, and (for visibilities with baseline *a*-*b*):

$$\mathcal{V}^{Sky}_{mod,RL}(u,v) = \sum_s (q_s + j u_s) \left[\sum_k I^s_k e^{2\pi j (u\alpha^s_k + v\delta^s_k)} \right] + \left((D^{Sky}_R)_a + (D^{Sky}_L)^*_b \right) \mathcal{V}^{Sky}_{obs,I}$$

and similarly for LR. The fitting parameters are q_s, u_s, and the Dterms (D_R and D_L for antennas a and b). All these parameters *are linear with* $\mathcal{V}^{Sky}_{mod,RL}$.

- Reduces the polarization calibration to a mere *linear least-squares* problem (i.e., just a matrix inversion, and that's it!).

The PolSolve Algorithm

- Fits the Dterms (and source polarization) using the full Meq (i.e., second-order approximation; good for very high fractional polarizations and/or Dterms). Uses either the slicing or the polarization selfcal approach.
- Computes the error function *in the Receiver Frame*.

$$\chi^2 = \sum_{i,pol} W_i \left(\mathcal{V}^{Rec}_{mod,pol} - \mathcal{V}^{Rec}_{obs,pol} \right)^2_i$$

where *pol* can be *RL* and *LR*, and (for visibilities with baseline *a*-*b*):

$$\mathcal{V}^{Rec}_{mod,RL}(u,v) = \left(\sum_s (q_s + j u_s) I^s_{mod} \right) (e^{-j\Delta}) + ((D_R)_a + (D_L)^*_b)\, \mathcal{V}^{Rec}_{obs,I} + (D_L)_a\, (D_R)^*_b\, \mathcal{V}^{Rec}_{obs,LR}$$

and similarly for LR. The fitting parameters are the same as with PolSolve.

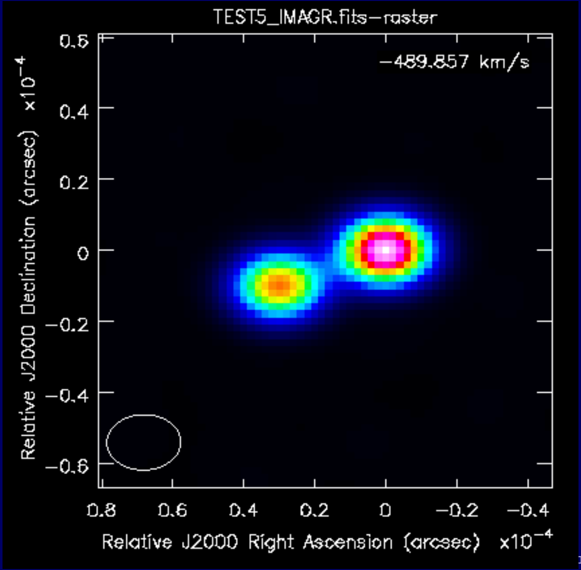
- Current developments: multi-source calibration, wide-band fitting, graphical interface (for sub-component editing), dealing with Stokes V, linear polarizers, etc.

PolSolve vs. LPCAL

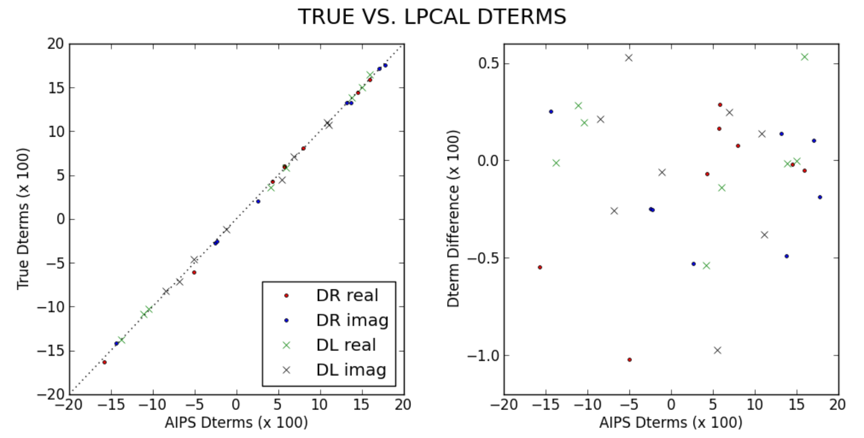
LPCAL vs. PolSolve (synthetic data)

- **Source** (double source):
  - **Comp #1:** 1.4 Jy with $m = 25.00\%$, EVPA $= 26.57^\circ$.
  - **Comp #2:** 1.0 Jy with $m = 28.43\%$, EVPA $= 70.36^\circ$.
  - **Coordinates**: J2000 12h30m49.4234 -12d23m28.0439
- **Observing frequency**: 230 GHz
- **Bandwidth**: 1 GHz (4 spectral channels).
- **On-source time**: 3 hr (10 scans of 0.3 h each); **Total time**: 12 h.
- **Stations** (eight EHT telescopes):
  - ALMA, APEX, Arizona, JCMT, LMT, SMA, SPT, PV.
- **Noise**: $\tau = 0.01$; $T_{sky} = 250\,\mathrm{K}$; $T_{ground} = 270\,\mathrm{K}$; $T_{rec} = 50\,\mathrm{K}$.
- **Dterms**: Random Gaussian amplitude distribution (centered on 7% with $\sigma = 2\%$) and random Uniform phase distribution.

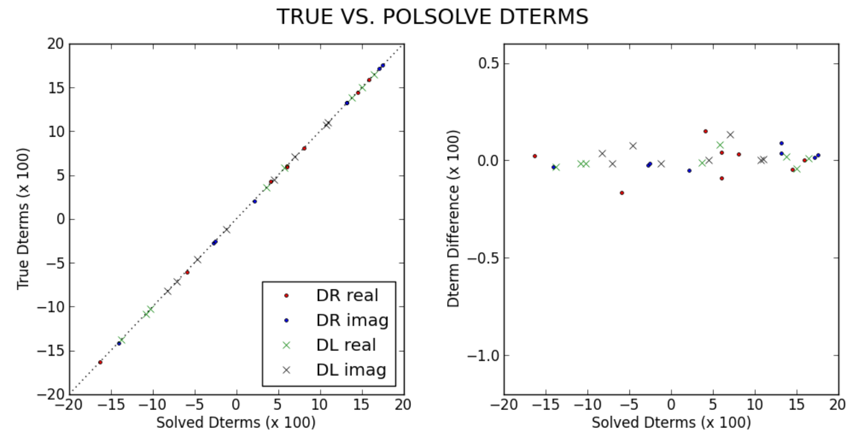
LPCAL vs. PolSolve (synthetic data)



LPCAL vs. PolSolve (synthetic data)



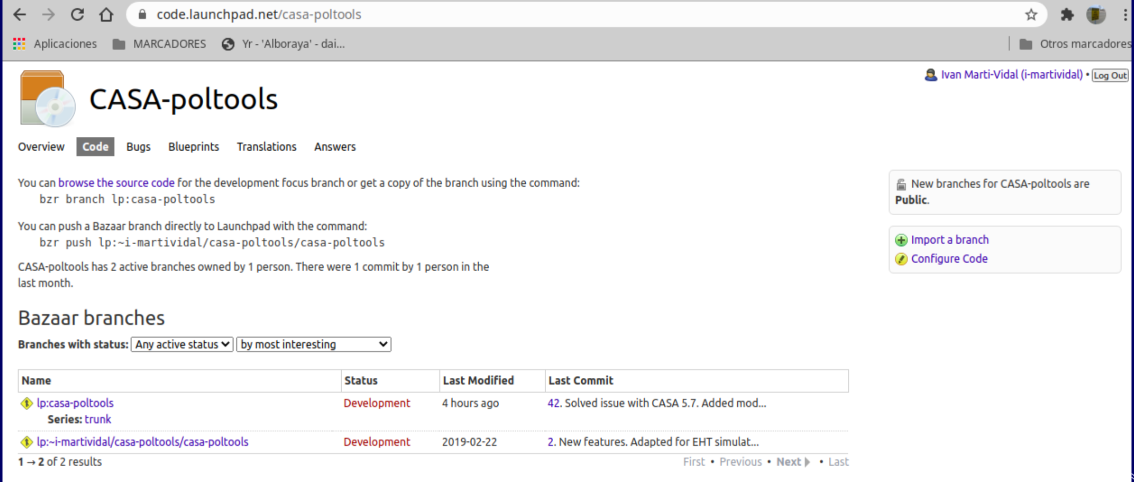
LPCAL vs. PolSolve (synthetic data)



Using PolSolve

PolSolve Installation

Download it:



code.launchpad.net/casa-poltools

Aplicaciones MARCADORES Yr - 'Alboraya' - dai... Otros marcadores

Ivan Marti-Vidal (i-martividal) • Log Out

CASA-poltools

Overview Code Bugs Blueprints Translations Answers

You can browse the source code for the development focus branch or get a copy of the branch using the command:

```
bzr branch lp:casa-poltools
```

You can push a Bazaar branch directly to Launchpad with the command:

```
bzr push lp:~i-martividal/casa-poltools/casa-poltools
```

CASA-poltools has 2 active branches owned by 1 person. There were 1 commit by 1 person in the last month.

New branches for CASA-poltools are **Public**.

Import a branch
Configure Code

Bazaar branches

Branches with status: Any active status | by most interesting

Name	Status	Last Modified	Last Commit
lp:casa-poltools **Series:** trunk	Development	4 hours ago	42. Solved issue with CASA 5.7. Added mod...
lp:~i-martividal/casa-poltools/casa-poltools	Development	2019-02-22	2. New features. Adapted for EHT simulat...

1 → 2 of 2 results First • Previous • **Next** • Last

PolSolve Installation

Compile and install:

```
####################################
### OPTION 1:

   3.2.- Just run in a terminal (from the PolSolve directory):

      rm -f *.so
      $CASABASE/bin/python setup.py build_ext --inplace

   (where $CASABASE is the path to your CASA installation).
```

```
#######################################

Now, it's time to load the task into CASA:

4.- Run "buildmytasks" in that directory:

    $CASAPATH/bin/buildmytasks


5.- Edit the file /home/you/.casa/init.py
    You may need to execute CASA at least once,
    to have that ".casa" directory with the proper content.
    In that "init.py", add the lines:

       import sys
       sys.path.append('/home/you/.casa/PolTools')
       execfile('/home/you/.casa/PolTools/mytasks.py')


6.- That's it! You should be able to run the new tasks in CASA!
    Just doing:

         task polsolve    or   tget polsolve
         task polsimulate    or   tget polsimulate


    should load the tasks.
```

PolSolve Interface: Keywords

```
---------> inp()
#  polsolve :: Version 1.0.1b - Leakage solver for circular polarizers and extended polarization calibrators.

vis                 =  'AUX2.ms'        #  Name of input measurement set. The
                                        #   data should already be calibrated
                                        #   (in bandpass and gains).
spw                 =        '0'        #  Spectral window(s) to be fitted. Can
                                        #   be an integer, a list of integers,
                                        #   or a CASA-like selection string.
                                        #   Default = all spws are used.
field               =      'M87'        #  Field name (or id) to use as
                                        #   calibrator. Follows CASA syntax for
                                        #   several fields.
mounts              = ['AZ', 'NR', 'NR', 'NL', 'AZ', 'NL', 'NL', 'NL', 'AZ'] #  L
                                        #   ist of the antenna mounts (must be
                                        #   given  in the same order as the
                                        #   ANTENNA table  of the ms). Use this
                                        #   in case the mounts  were not
                                        #   properly imported into the ms. A
                                        #   mount type is specified with two
                                        #   characters. Supported mounts:  alt-
                                        #   az ('AZ'),  equatorial ('EQ'),  X-Y
                                        #   ('XY'),  Nasmyth Right ('NR') and
                                        #   Nasmyth Left ('NL').  Default means
                                        #   all antennas are alt-az.
feed_rotation       =         []        #  Rotation of the feed of each antenna
                                        #   with respect to the local
                                        #   horizontal-vertical frame. One value
                                        #   per antenna. Empty list assumes a
                                        #   null feed angle for all antennas.
```

PolSolve Interface: Keywords

```
DR          =         []        #  List of complex numbers (length equal
                                #   to the number of antennas). If not
                                #   empty,   these are the a-priori
                                #   values of   the DR leakage terms to
                                #   use   in the fit. The name of a
                                #   file, with the list pickled, can
                                #   also be given.
DL          =         []        #  List of complex numbers (length equal
                                #   to the number of antennas). If not
                                #   empty,   these are the a-priori
                                #   values of   the DL leakage terms to
                                #   use   in the fit. The name of a
                                #   file, with the list pickled, can
                                #   also be given.
DRSolve     = [True, True, False, False, True, False, True, False, False] #  L
                                #   ist of booleans (length equal to the
                                #   number of antennas). If not empty,
                                #   it will tell which DR terms are
                                #   fitted.   The DR[i] term will be
                                #   fitted if   DRSolve[i] is True.
                                #   Otherwise, DR[i]   will be fixed to
                                #   its a-priori value.   Default (i.e.,
                                #   empty list) means   to fit all DRs.
DLSolve     = [True, True, False, False, True, False, True, False, False] #  J
                                #   ust as DRSolve, but for the DL
                                #   terms
```

PolSolve Interface: Keywords

```
CLEAN_models      =     [[1.0]]          #  List of CLEAN model files (CCs given
                                         #   in PRTAB format). Each file will
                                         #   correspond to a source component
                                         #   with   the same polarization state.
                                         #   If one number   is given (instead of
                                         #   a list of filenames),  a centered
                                         #   point source (with that   flux
                                         #   density) will be used.  If more than
                                         #   one field is fitted, CLEAN_models
                                         #   will  be a List of Lists (with one
                                         #   list per field).   If names of CASA
                                         #   IQU[V] model image(s) are   given
                                         #   (i.e., filenames ended   with
                                         #   ".model"), these   models will be
                                         #   used and FIXED (i.e.,   Pfrac, EVPA
                                         #   and PolSolve will NOT be used).   If
                                         #   equal to "model_column", the model
                                         #   column   (for all correlation
                                         #   products) will be used.
```

PolSolve Interface: Keywords

```
Pfrac          =     [[0.0]]        # List of fractional polarizations (one
                                    #  number   per source component).
                                    #  Pfrac    values must fall between 0
                                    #  and 1.  If more than one field is
                                    #  fitted,   this will be a List of
                                    #  Lists (one list per field).
EVPA           =     [[0.0]]        # List of EVPAs in degrees (one number
                                    #  per source component). Angles   are
                                    #  measured from North to East.  If
                                    #  more than one field is fitted,
                                    #  this will be a List of Lists (one
                                    #  list per field).
PolSolve       =    [[True]]        # List of booleans (one per source
                                    #  component)   that tell which source
                                    #  components   are to be fitted in
                                    #  polarization.   If PolSolve[i] is
                                    #  True, the fractional   polarization
                                    #  and EVPA of the ith source
                                    #  component will be fitted, together
                                    #  with   the antenna Dterms. If False,
                                    #  all   Stokes parameters of the ith
                                    #  component will   be fixed in the
                                    #  fit.   Empty list means to fit the
                                    #  polarization   of all the source
                                    #  components.  If more than one field
                                    #  is fitted,   this will be a List of
```

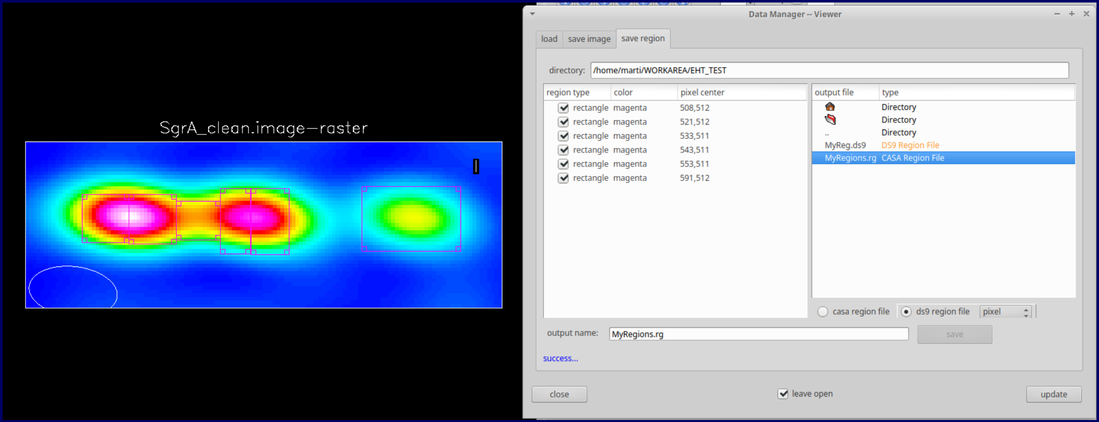
PolSolve Interface: Keywords


```
parang_corrected    =     True        #  If True, the data are assumed to be
                                      #   already corrected for parallactic
                                      #   angle.   This is usually the case,
                                      #   unless   you are working with data
                                      #   generated with   polsimulate with no
                                      #   parang correction.
target_field        =      ''         #  List of sources to which apply the
                                      #   Dterm (and parangle) correction. It
                                      #   must follow the CASA syntax if a
                                      #   range of field ids is given. Empy
                                      #   list means NOT to apply the Dterms
                                      #   (i.e., just save them in a
                                      #   calibration table). If you want to
                                      #   apply the calibration, DO NOT FORGET
                                      #   TO *ALWAYS* RUN CLEARCAL BEFORE
                                      #   POLSOLVE!!
```

PolSolve Interface: Keywords

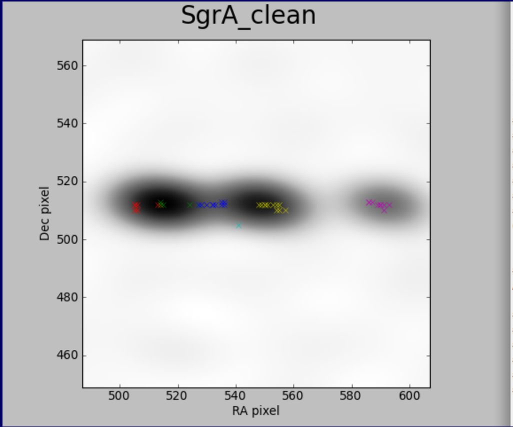
```
plot_parang         =    True      # If True, plot the time evolution of
                                   #  the antenna feed angles (i.e.,
                                   #  parallactic angle plus correction
                                   #  from the antenna mounts).
min_elev_plot       =    10.0      #  In degrees. If plot_parang is True,
                                   #  points with elevations lower than
                                   #  this limit will be plotted in red.
                                   #  THIS DOES NOT FLAG THE DATA. If you
                                   #  want to flag them, run the flagdata
                                   #  task.
wgt_power           =     1.0      #  Power for the visibility weights.
                                   #  Unity means to leave the weights
                                   #  untouched (i.e., equivalent to
                                   #  natural weighting, but for the fit).
                                   #  Zero means equal weights for all
                                   #  visibilities (i.e., equivalent to
                                   #  uniform weighting for the fit).
rewgt_pfrac         =     0.0      # Modify the weights by
                                   #  1/(1+rewgt_pfrac*p), where p is the
                                   #  source contribution to the
                                   #  visibility fractional polarization
                                   #  (estimated from the best-fit model
                                   #  with the original weights).
linear_approx       =   False      # If True, solve the polarimetry by
                                   #  assuming linear dependence of
                                   #  visibilities with Dterms (i.e., a la
                                   #  LPCAL; faster).
plot_residuals      =    True      # If True, plot the residual cross-hand
                                   #  visibilities vs. difference of
                                   #  parallactic angle.
```

PolSolve Interface: Slicing approach



First, run (t)`CLEAN`, set the regions (e.g., with `imview`) and save them into a file. and estimate de Dterms

PolSolve Interface: Slicing approach




```python
import numpy as np
import pylab as pl
import os, sys
import glob

# The user should use the task "imview" to get the pixel coordinates of the
# BLC and TRC of the boxes that will define the polarization sub-components.
# In this case, we are using several sub-components, stored in the "MyRegions.rg"
# file. The helper task called "CCextract" takes all the CASA (or DS9) regions
# defined there (you can use  boxes, circles, ellipses, etc.), extracts the
# CC components of each region and saves them in ascii format, to be read by
# PolSolve:
CCextract(model_image='SgrA_clean.model',
          regions='MyRegions.rg',make_plot=True)

## Get all the subcomponent files:
ALLCCs = glob.glob('SgrA_clean.model.CC??')

# NOTE: CCextract has an option to make a plot of the CLEAN deltas in each
# sub-component (using different colors). In addition, you should pay attention
# to the warnings regarding CLEAN components appearing in more than one region.
# If this happens, the CLEAN component is only added to the region first
# appearing in the list. This behaviour can indeed be used to make fancy
# boolean region conditions!
```

Second, execute `CCextract` (generate CC files from the regions). and estimate de Dterms

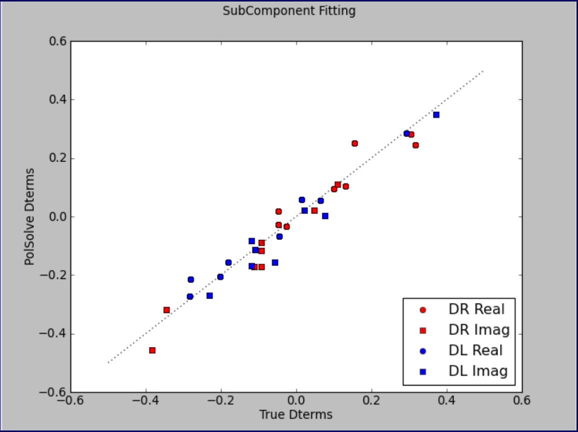
PolSolve Interface: Slicing approach

```
Undoing parang correction
 ITER 7. ChiSq 7.084e-04 ; Model Factors:  0 -> 1.000e+00
Fitting results:
FIELD ID 0:

Source id 0 (POLSIM), Component #0: Pfrac =  1.1783 +/- 0.006 ; EVPA =  -3.28 +/- 0.15 deg. (Q/I = 1.170e+00 +/- 6.33
Source id 0 (POLSIM), Component #1: Pfrac =  0.0607 +/- 0.001 ; EVPA =  34.84 +/- 0.41 deg. (Q/I = 2.107e-02 +/- 8.82
Source id 0 (POLSIM), Component #2: Pfrac =  0.5079 +/- 0.004 ; EVPA = -66.59 +/- 0.23 deg. (Q/I = -3.473e-01 +/- 3.9
Source id 0 (POLSIM), Component #3: Pfrac =  0.5037 +/- 0.002 ; EVPA =  22.29 +/- 0.09 deg. (Q/I = 3.588e-01 +/- 1.52
Source id 0 (POLSIM), Component #4: Pfrac =  28.9944 +/- 0.206 ; EVPA =  30.28 +/- 0.22 deg. (Q/I = 1.425e+01 +/- 2.1
Source id 0 (POLSIM), Component #5: Pfrac =  0.2301 +/- 0.001 ; EVPA = -48.12 +/- 0.10 deg. (Q/I = -2.498e-02 +/- 7.9

 Dterms (Right):
Antenna #0 (AA):  Real =  9.58e-02 +/- 5.7e-04; Imag = -8.85e-02 +/- 5.7e-04 | Amp = 0.1304 | Phase: -42.71 deg.
Antenna #1 (AP):  Real = -3.49e-02 +/- 6.2e-04; Imag =  1.10e-01 +/- 6.2e-04 | Amp = 0.1151 | Phase: 107.67 deg.
Antenna #2 (AZ):  Real =  1.05e-01 +/- 1.6e-03; Imag = -1.15e-01 +/- 1.6e-03 | Amp = 0.1560 | Phase: -47.58 deg.
Antenna #3 (JC):  Real =  2.81e-01 +/- 1.2e-03; Imag = -1.71e-01 +/- 1.2e-03 | Amp = 0.3287 | Phase: -31.40 deg.
Antenna #4 (LM):  Real = -2.62e-02 +/- 5.8e-04; Imag =  2.06e-02 +/- 5.8e-04 | Amp = 0.0333 | Phase: 141.93 deg.
Antenna #5 (SM):  Real =  1.89e-02 +/- 2.6e-03; Imag = -4.55e-01 +/- 2.5e-03 | Amp = 0.4550 | Phase: -87.62 deg.
Antenna #6 (SP):  Real =  2.46e-01 +/- 1.3e-03; Imag = -3.17e-01 +/- 1.3e-03 | Amp = 0.4015 | Phase: -52.21 deg.
Antenna #7 (PV):  Real =  2.51e-01 +/- 1.5e-03; Imag = -1.73e-01 +/- 1.5e-03 | Amp = 0.3049 | Phase: -34.55 deg.

 Dterms (Left):
Antenna #0 (AA):  Real = -2.06e-01 +/- 5.7e-04; Imag = -1.12e-01 +/- 5.7e-04 | Amp = 0.2346 | Phase: -151.46 deg.
Antenna #1 (AP):  Real =  5.65e-02 +/- 6.5e-04; Imag =  2.01e-02 +/- 6.5e-04 | Amp = 0.0600 | Phase: 19.62 deg.
Antenna #2 (AZ):  Real = -1.57e-01 +/- 1.6e-03; Imag = -2.70e-01 +/- 1.6e-03 | Amp = 0.3125 | Phase: -120.07 deg.
Antenna #3 (JC):  Real = -2.13e-01 +/- 1.2e-03; Imag =  4.01e-03 +/- 1.2e-03 | Amp = 0.2133 | Phase: 178.92 deg.
Antenna #4 (LM):  Real =  2.86e-01 +/- 6.3e-04; Imag = -1.68e-01 +/- 6.4e-04 | Amp = 0.3314 | Phase: -30.41 deg.
Antenna #5 (SM):  Real = -6.82e-02 +/- 2.6e-03; Imag = -1.55e-01 +/- 2.6e-03 | Amp = 0.1694 | Phase: -113.74 deg.
Antenna #6 (SP):  Real =  5.94e-02 +/- 1.2e-03; Imag = -8.12e-02 +/- 1.2e-03 | Amp = 0.1006 | Phase: -53.80 deg.
Antenna #7 (PV):  Real = -2.71e-01 +/- 1.2e-03; Imag =  3.50e-01 +/- 1.2e-03 | Amp = 0.4432 | Phase: 127.74 deg.

CASA <7>:
```

```
# Solve for Dterms and source polarization:
polsolve(vis='SgrA_polsimulate.ms',
## The mounts should be given following the order of the ANTENNA table:
         mounts=['AZ', 'NR', 'NR', 'AZ', 'NL', 'NL', 'AZ', 'NL'],
## This is the list of ascii files created by "CCextract":
         CLEAN_models=ALLCCs,
## Initial guesses of the polarization state of each sub-component:
         Pfrac=[0. for pi in ALLCCs], EVPA=[0. for pi in ALLCCs],
## Solve for the polarization of all three sub-components:
PolSolve=[True for pi in ALLCCs],
## Apply the full non-linear Dterm model:
linear_approx=False)

## Change the name of the Dterm CASA table, to keep it:
os.system('rm -rf LPCAL.Dterms')
os.system('cp -r SgrA_polsimulate.ms.spw_0.Dterms LPCAL.Dterms')

## READ FITTED DTERMS:
tb.open('SgrA_polsimulate.ms.spw_0.Dterms')
DTS = tb.getcol('CPARAM')
tb.close()



## RECOVER SIMULATED DTERMS:
```

Finally, use the CC files with `polsolve` and estimate de Dterms

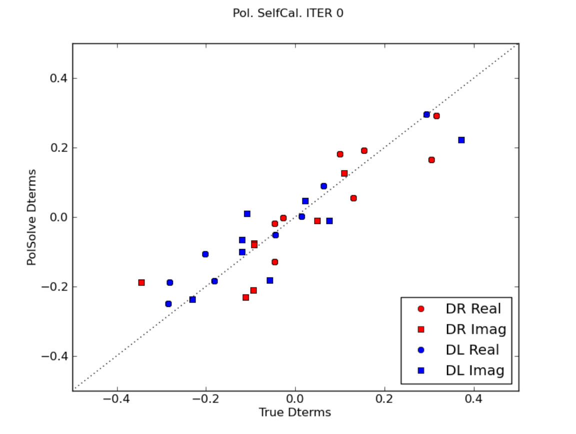
PolSolve Interface: Slicing approach



Finally, use the CC files with `polsolve` and estimate de Dterms

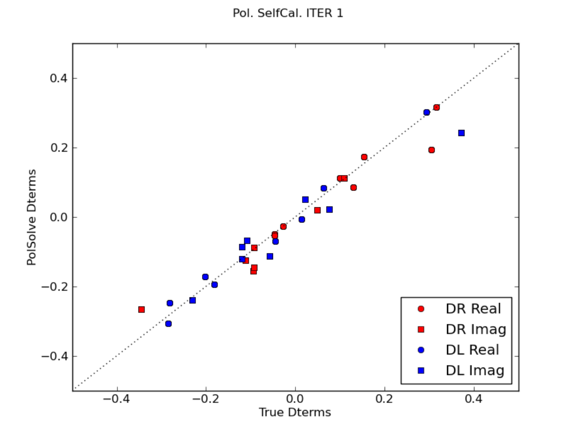
PolSolve Interface: Pol. SelfCal

```
# ITERATE!!
for i in range(NITER):

  print '\n\n ITERATION %i\n\n'%i

## We CLEAN the image, using the visibilities stored in
# the "corrected" column of the measurement set:
  os.system('rm -rf SgrA_SC_clean*')
  tclean(vis='SgrA_polsimulate.ms',
    imagename='SgrA_SC_clean',
    specmode = 'mfs',
    niter = 200,
    interactive=False,
    imsize=1024,
    cell = '0.000001arcsec',
    stokes = 'IQUV',
    mask = 'TCLEAN.mask', ## We use the mask that we created manually.
    weighting = 'uniform',
    deconvolver='hogbom',
    gain=0.1,
    restart=False)

# We remove the previous Dterm table:
  os.system('rm -rf SgrA_polsimulate.ms.spw_0.Dterms')

# Solve for the Dterms:
  polsolve(vis='SgrA_polsimulate.ms',
## If we set "target_field", the Dterms will not only be
# estimated, but also APPLIED to the "corrected" column.
# This is ESSENTIAL for the self-cal approach to work:
        target_field='POLSIM', # Source name
## Antenna mounts (the ordering should be that of the ANTENNA table):
        mounts=['AZ', 'NR', 'NR', 'AZ', 'NL', 'NL', 'AZ', 'NL'],
## If "CLEAN_models" is the path to a (full-polarization) CASA image,
## the self-calibration approach is activated:
        CLEAN_models='SgrA_SC_clean.model',
# These values are not really used:
        Pfrac=[0.], EVPA=[0.], PolSolve=[False],
## Set this to True, if you want to test the linear Dterm model:
        linear_approx=False)
```



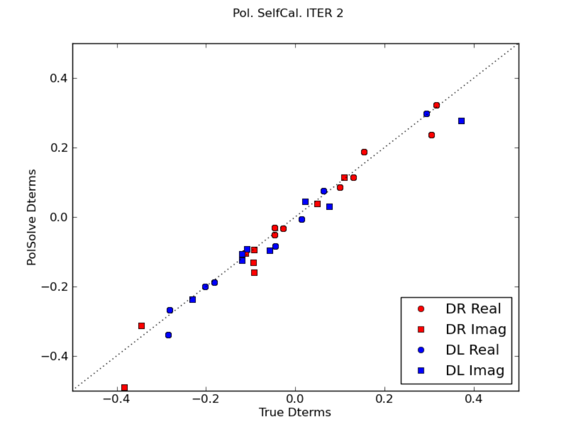
PolSolve Interface: Pol. SelfCal

```
# ITERATE!!
for i in range(NITER):

  print '\n\n ITERATION %i\n\n'%i

## We CLEAN the image, using the visibilities stored in
# the "corrected" column of the measurement set:
  os.system('rm -rf SgrA_SC_clean*')
  tclean(vis='SgrA_polsimulate.ms',
    imagename='SgrA_SC_clean',
    specmode = 'mfs',
    niter = 200,
    interactive=False,
    imsize=1024,
    cell = '0.000001arcsec',
    stokes = 'IQUV',
    mask = 'TCLEAN.mask', ## We use the mask that we created manually.
    weighting = 'uniform',
    deconvolver='hogbom',
    gain=0.1,
    restart=False)

# We remove the previous Dterm table:
  os.system('rm -rf SgrA_polsimulate.ms.spw_0.Dterms')

# Solve for the Dterms:
  polsolve(vis='SgrA_polsimulate.ms',
## If we set "target_field", the Dterms will not only be
# estimated, but also APPLIED to the "corrected" column.
# This is ESSENTIAL for the self-cal approach to work:
        target_field='POLSIM', # Source name
## Antenna mounts (the ordering should be that of the ANTENNA table):
        mounts=['AZ', 'NR', 'NR', 'AZ', 'NL', 'NL', 'AZ', 'NL'],
## If "CLEAN_models" is the path to a (full-polarization) CASA image,
## the self-calibration approach is activated:
        CLEAN_models='SgrA_SC_clean.model',
# These values are not really used:
        Pfrac=[0.], EVPA=[0.], PolSolve=[False],
## Set this to True, if you want to test the linear Dterm model:
        linear_approx=False)
```



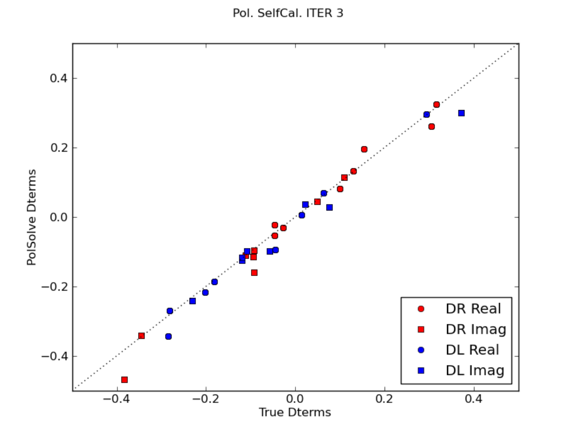
PolSolve Interface: Pol. SelfCal

```
# ITERATE!!
for i in range(NITER):

  print '\n\n ITERATION %i\n\n'%i

## We CLEAN the image, using the visibilities stored in
# the "corrected" column of the measurement set:
  os.system('rm -rf SgrA_SC_clean*')
  tclean(vis='SgrA_polsimulate.ms',
    imagename='SgrA_SC_clean',
    specmode = 'mfs',
    niter = 200,
    interactive=False,
    imsize=1024,
    cell = '0.000001arcsec',
    stokes = 'IQUV',
    mask = 'TCLEAN.mask', ## We use the mask that we created manually.
    weighting = 'uniform',
    deconvolver='hogbom',
    gain=0.1,
    restart=False)

# We remove the previous Dterm table:
  os.system('rm -rf SgrA_polsimulate.ms.spw_0.Dterms')

# Solve for the Dterms:
  polsolve(vis='SgrA_polsimulate.ms',
## If we set "target_field", the Dterms will not only be
# estimated, but also APPLIED to the "corrected" column.
# This is ESSENTIAL for the self-cal approach to work:
        target_field='POLSIM', # Source name
## Antenna mounts (the ordering should be that of the ANTENNA table):
        mounts=['AZ', 'NR', 'NR', 'AZ', 'NL', 'NL', 'AZ', 'NL'],
## If "CLEAN_models" is the path to a (full-polarization) CASA image,
## the self-calibration approach is activated:
        CLEAN_models='SgrA_SC_clean.model',
# These values are not really used:
        Pfrac=[0.], EVPA=[0.], PolSolve=[False],
## Set this to True, if you want to test the linear Dterm model:
        linear_approx=False)
```



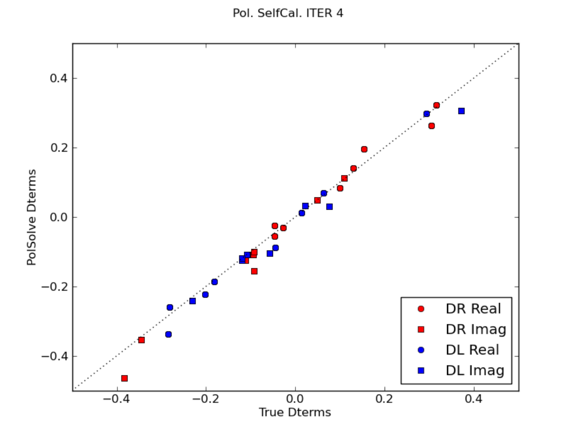
PolSolve Interface: Pol. SelfCal

```
# ITERATE!!
for i in range(NITER):

  print '\n\n ITERATION %i\n\n'%i

## We CLEAN the image, using the visibilities stored in
# the "corrected" column of the measurement set:
  os.system('rm -rf SgrA_SC_clean*')
  tclean(vis='SgrA_polsimulate.ms',
    imagename='SgrA_SC_clean',
    specmode = 'mfs',
    niter = 200,
    interactive=False,
    imsize=1024,
    cell = '0.000001arcsec',
    stokes = 'IQUV',
    mask = 'TCLEAN.mask', ## We use the mask that we created manually.
    weighting = 'uniform',
    deconvolver='hogbom',
    gain=0.1,
    restart=False)

# We remove the previous Dterm table:
  os.system('rm -rf SgrA_polsimulate.ms.spw_0.Dterms')

# Solve for the Dterms:
  polsolve(vis='SgrA_polsimulate.ms',
## If we set "target_field", the Dterms will not only be
# estimated, but also APPLIED to the "corrected" column.
# This is ESSENTIAL for the self-cal approach to work:
        target_field='POLSIM', # Source name
## Antenna mounts (the ordering should be that of the ANTENNA table):
        mounts=['AZ', 'NR', 'NR', 'AZ', 'NL', 'NL', 'AZ', 'NL'],
## If "CLEAN_models" is the path to a (full-polarization) CASA image,
## the self-calibration approach is activated:
        CLEAN_models='SgrA_SC_clean.model',
# These values are not really used:
        Pfrac=[0.], EVPA=[0.], PolSolve=[False],
## Set this to True, if you want to test the linear Dterm model:
        linear_approx=False)
```



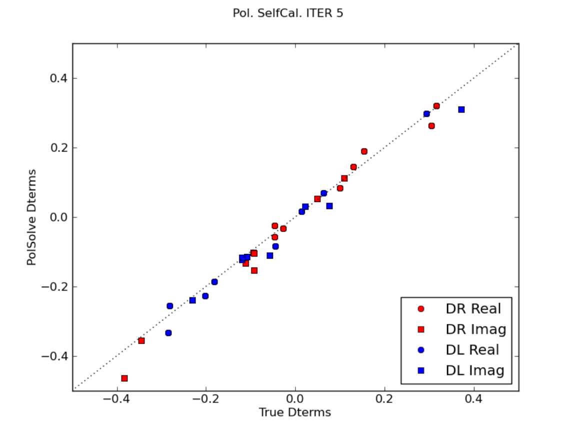
PolSolve Interface: Pol. SelfCal

```
# ITERATE!!
for i in range(NITER):

  print '\n\n ITERATION %i\n\n'%i

## We CLEAN the image, using the visibilities stored in
# the "corrected" column of the measurement set:
  os.system('rm -rf SgrA_SC_clean*')
  tclean(vis='SgrA_polsimulate.ms',
    imagename='SgrA_SC_clean',
    specmode = 'mfs',
    niter = 200,
    interactive=False,
    imsize=1024,
    cell = '0.000001arcsec',
    stokes = 'IQUV',
    mask = 'TCLEAN.mask', ## We use the mask that we created manually.
    weighting = 'uniform',
    deconvolver='hogbom',
    gain=0.1,
    restart=False)

# We remove the previous Dterm table:
  os.system('rm -rf SgrA_polsimulate.ms.spw_0.Dterms')

# Solve for the Dterms:
  polsolve(vis='SgrA_polsimulate.ms',
## If we set "target_field", the Dterms will not only be
# estimated, but also APPLIED to the "corrected" column.
# This is ESSENTIAL for the self-cal approach to work:
        target_field='POLSIM', # Source name
## Antenna mounts (the ordering should be that of the ANTENNA table):
        mounts=['AZ', 'NR', 'NR', 'AZ', 'NL', 'NL', 'AZ', 'NL'],
## If "CLEAN_models" is the path to a (full-polarization) CASA image,
## the self-calibration approach is activated:
        CLEAN_models='SgrA_SC_clean.model',
# These values are not really used:
        Pfrac=[0.], EVPA=[0.], PolSolve=[False],
## Set this to True, if you want to test the linear Dterm model:
        linear_approx=False)
```

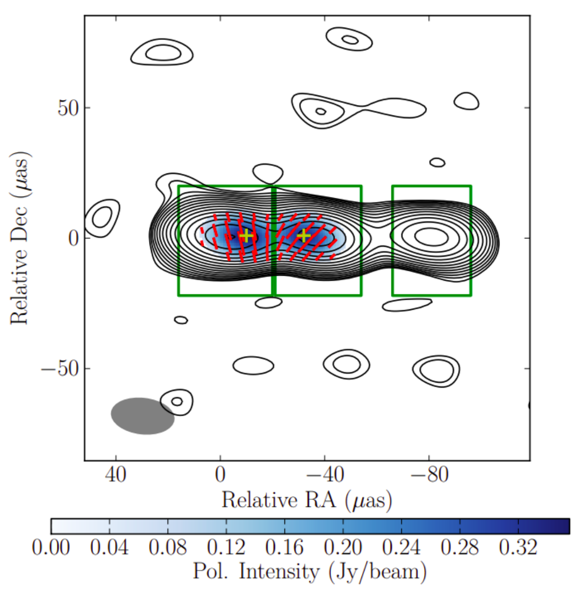


PolSolve Interface: Pol. SelfCal

```
# ITERATE!!
for i in range(NITER):

  print '\n\n ITERATION %i\n\n'%i

## We CLEAN the image, using the visibilities stored in
# the "corrected" column of the measurement set:
  os.system('rm -rf SgrA_SC_clean*')
  tclean(vis='SgrA_polsimulate.ms',
    imagename='SgrA_SC_clean',
    specmode = 'mfs',
    niter = 200,
    interactive=False,
    imsize=1024,
    cell = '0.000001arcsec',
    stokes = 'IQUV',
    mask = 'TCLEAN.mask', ## We use the mask that we created manually.
    weighting = 'uniform',
    deconvolver='hogbom',
    gain=0.1,
    restart=False)

# We remove the previous Dterm table:
  os.system('rm -rf SgrA_polsimulate.ms.spw_0.Dterms')

# Solve for the Dterms:
  polsolve(vis='SgrA_polsimulate.ms',
## If we set "target_field", the Dterms will not only be
# estimated, but also APPLIED to the "corrected" column.
# This is ESSENTIAL for the self-cal approach to work:
        target_field='POLSIM', # Source name
## Antenna mounts (the ordering should be that of the ANTENNA table):
        mounts=['AZ', 'NR', 'NR', 'AZ', 'NL', 'NL', 'AZ', 'NL'],
## If "CLEAN_models" is the path to a (full-polarization) CASA image,
## the self-calibration approach is activated:
        CLEAN_models='SgrA_SC_clean.model',
# These values are not really used:
        Pfrac=[0.], EVPA=[0.], PolSolve=[False],
## Set this to True, if you want to test the linear Dterm model:
        linear_approx=False)
```



And finally...



50
0
−50
Relative Dec (μas)
40
0
−40
−80
Relative RA (μas)
0.00 0.04 0.08 0.12 0.16 0.20 0.24 0.28 0.32
Pol. Intensity (Jy/beam)

SUMMARY

- Instrumental polarization can be decoupled from source polarization thanks to the Earh rotation. We *need* it to calibrate our data.

- Polarization calibration in VLBI is tricky (structure effects from the calibrators). Several approaches to account for it:
  - Inverse modelling (i.e., " à la CLEAN "). The only (known) option for CASA is `polsolve`. Other options (for AIPS/Difmap) are `LPCAL` and `GPCAL`.
  - Forward modelling (i.e., " à la MEM "). E.g., `EHTim`.
  - MCMC methods (i.e., " à la brute-force "). E.g., `THEMIS` and `DMC`.

- `polsolve` is an unofficial CASA task, which applies the calibration directly into the CORRECTED data column (due to CASA limitations in the handling of antenna mounts).
  This is a highly non-standard procedure.

- Several options available: subcomponent fitting (e.g., LPCAL approach) or polarization self-calibration (see, e.g., Cotton 1993).

THANKS TO OUR SPONSORS:

CASA VLBI

JUMPING JIVE
Joint Institute for VLBI
ERIC

RadioNet

THIS EVENT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON 2020 RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENTS 730562 (RADIONET) AND 7308844 (JUMPING JIVE)