JSkyMap

Map making for transit interferometers

R. Ansari &
C. Magneville , J.E. Campagne
& J. Zhang

July 2017

JSkyMap : Map making software (I)

(J. Zhang PhD)

- Reconstruct map from cleaned/calibrated visibilities
- Can handle data from dish arrays or cylinders, in transit mode
- Array geometry and feed/antenna beam should be known
- Can compute visibilities given an input sky map and set of baselines
- Written in C++, with parallel, multi-thread capability, over a single node, with mutiple CPU's or cores
  - Computation of beams in (l,m) space in parallel
  - Map reconstruction for different m-modes in parallel
- Can be used as a tool for the calibration stages
- Independent processing of different frequencies (can be handled in parallel on different nodes)

J. Zhang, R. Ansari, MNRAS 461, (2016) - arXiv:1606.03090

JSkyMap : Map making software (II)

(J. Zhang PhD)

- Do not yet handle polarisation, but extension is rather easy
- Except for the computation of polarised beam responses…
- The code is rather simple, built around few classes, but relies on the SOPHYA class library (**http://www.sophya.org**)
- Main classes used in JSkyMap :
  - BeamTP , BeamLM and BeamVis
  - SphCoordTrans , PseudoInverse<T>
  - JSphSkyMap
  - JSkyMap and BeamUV for planar geometry
  - Some utility functions

GIT repo: **https://gitlab.in2p3.fr/SCosmoTools/JSkyMap**

JSphSkyMap class

```
JSphSkyMap(int lmax=512, int m=256);
```

Computing visibility array from an input map

```
/* ==== compute visibilities in spherical geometry
    -Input :  Input sky map
              BeamLM list
              lmax (optional, if zero, use the value already in the class)
    -Output : The visibility matrix, with X-index corresponding to m-modes (0<=m<=SizeX())
              and Y-index corresponding to the beams ( SizeY() = 2*beams.size() )
              the factor two comes from the fact that visibilities for positive and negative m-modes
              are written as two separate rows of the array  */
TArray< complex<double> > ComputeVisibilities(
       SphereHEALPix<double> const& inmap,
       vector< BeamLM > const& beams, int lmax=0);
```

Computing map from a visibility array

```
/* ==== reconstructing map from a single set of visibilities
  -Input :
     visarr : one array of m-mode visibilities
     beams: list of beams
     wnoisecov == true , use the noise covariance matrix when inverting the A matrix to extract alm
     compcovar == true : Compute the error covariance matrix on estimated stky alm and save it to the PPF file
     compainva == true ; save A, Ainv Ainv * A matrices
     nthreads : nb of computing threads
   -Output :
     the returned JSphSkyMap object  */
JSphSkyMap ReconstructFromVisibilityArray(
     TArray<complex<double> >  & visarr,
     vector< BeamLM > const& beams,
     bool wnoisecov, bool compcovar, bool compainva, int nthreads=1);
```

JSkyMap : some utiliy programs

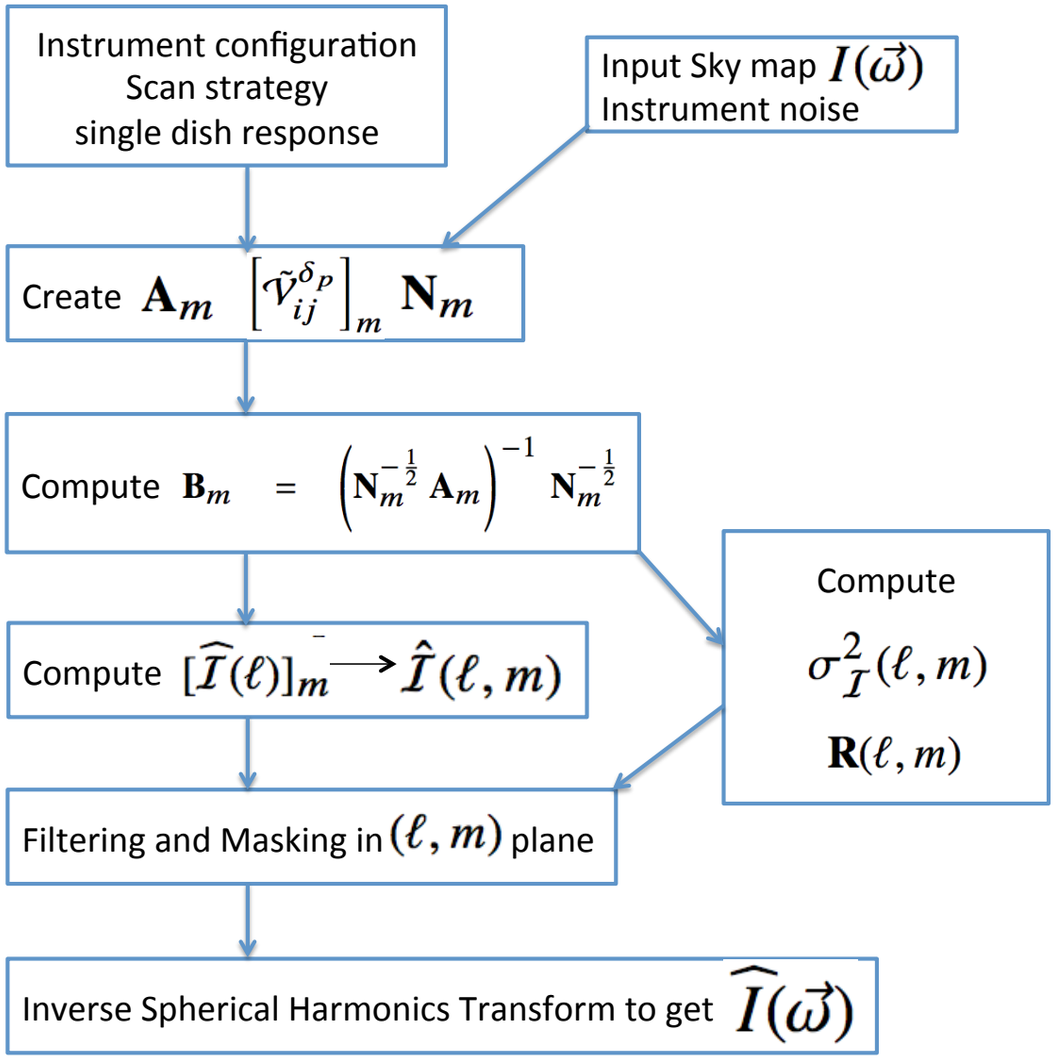
Computing visibility array from an input map (map2vis.cc)

```
 ------ map2vis.cc : Computing Visibility array from an input map and a set of
baselines -------
 map2vis/usage: map2vis InMapPPF_File OutPPF_File elevation baseline1  [baseline2
baseline3 ...]
  o elevation : elevation angle in degree (offset with respect to zenith in NS plane,
+ toward N
  o baselineS : baselineX,baselineY,baselineZ
```

Computing map from a visibility array (vis2map.cc)

```
------ vis2map.cc : reconstructing map from a set of visibility arrays -------
vis2map/usage: vis2map OutputMapPPF elevations VisiPPF1 [ VisiPPF2 ... ]
 o elevations : comma separated elevation angle values in degree
       (NS plane, + toward N)
 o VisiPPFS : Input visibility arrays
```

JSkyMap : Map making software (III)



Non polarised

$$\tilde{\mathcal{V}}_{ij}(m) = \sum_{\ell=|m|}^{+\ell_{\max}} (-1)^m\, \mathcal{I}(\ell,m)\mathcal{L}_{ij}(\ell,-m)$$

$$\tilde{\mathcal{V}}^*_{ij}(-m) = \sum_{\ell=|m|}^{+\ell_{\max}} \mathcal{I}(\ell,m)\mathcal{L}^*_{ij}(\ell,m)$$

$$\left[\tilde{\mathcal{V}}\right]_m = \mathbf{L}_m \times [\mathcal{I}(\ell)]_m + [\tilde{n}]_m$$

Polarised

$$\tilde{\mathcal{V}}_{p_i p_j}(m) = \sum_{\ell=|m|}^{+\ell_{max}} \sum_{\mathcal{X}} (-1)^m \mathcal{L}^X_{p_i p_j;l,-m}\mathcal{X}_{\ell m}$$

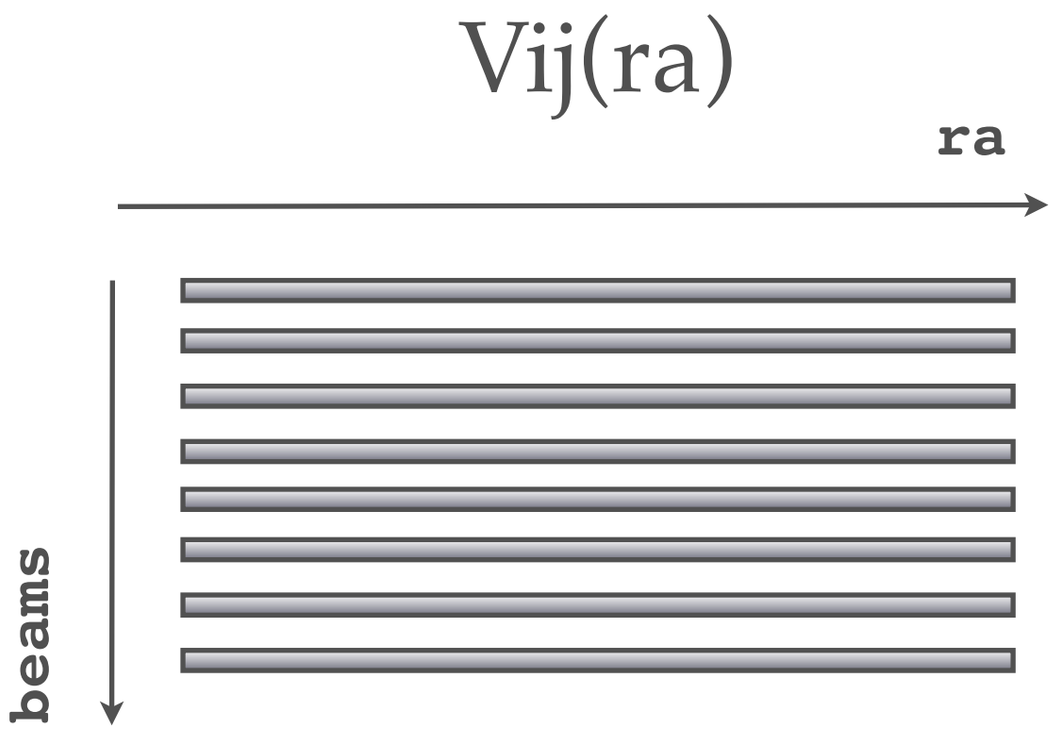
$$\tilde{\mathcal{V}}^*_{p_i p_j}(-m) = \sum_{\ell=|m|}^{+\ell_{max}} \sum_{\mathcal{X}} \mathcal{L}^{X*}_{p_i p_j;l,m}\mathcal{X}_{\ell m}$$

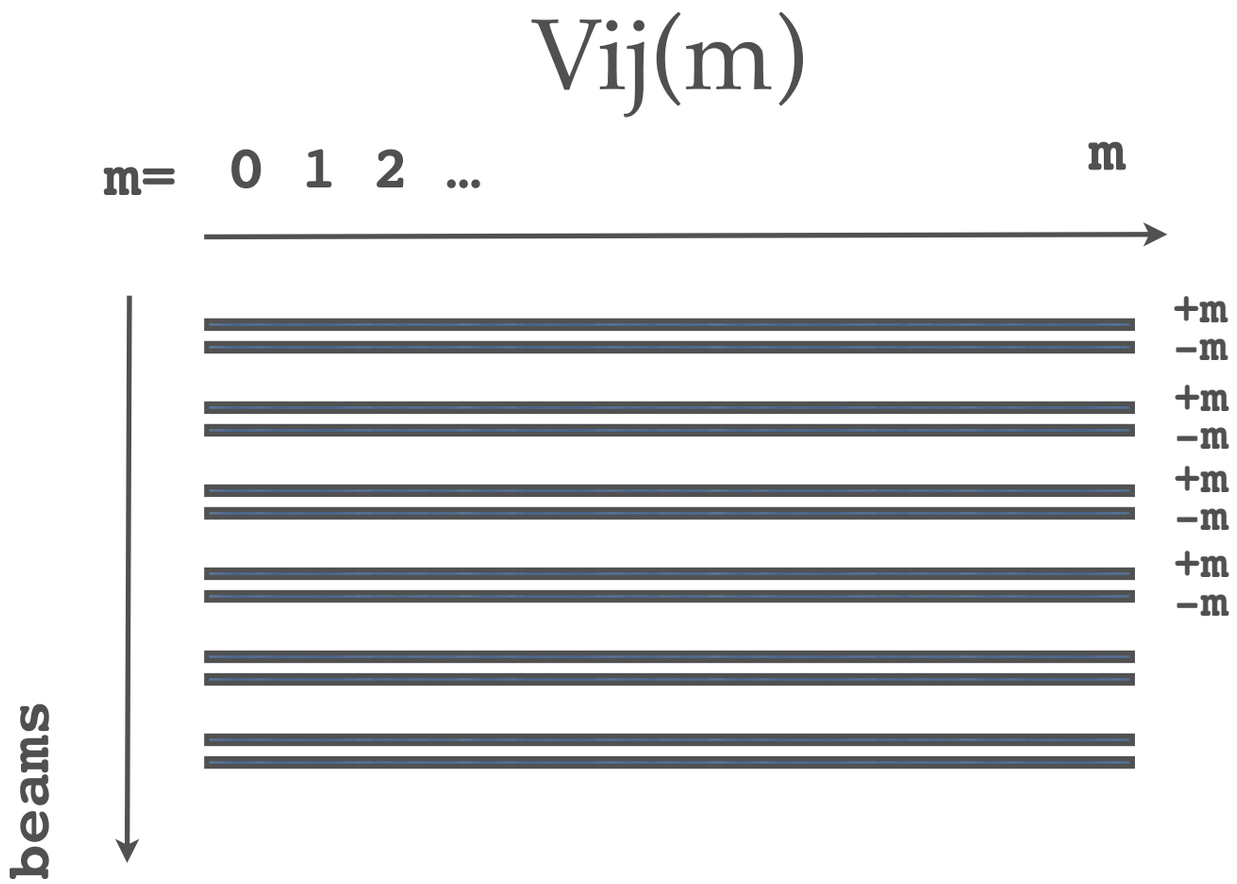
$$\mathcal{X} = \mathcal{I}, \mathcal{E}, \mathcal{B}, \mathcal{V}.$$

$$\mathcal{V}_{p_i,p_j} = \left[\mathcal{V}^{xx}_{ij}; \mathcal{V}^{yy}_{ij}; \mathcal{V}^{xy}_{ij}; \mathcal{V}^{yx}_{ij}\right]$$

$$p_i = \{(i,x),(i,y)\} \qquad p_j = \{(j,x),(j,y)\}$$

JSkyMap : Visibility array organisation





SOPHYA

A C++ class library for intensive data analysis and scientific computing

http://www.sophya.org

https://gitlab.in2p3.fr/SOPHYA