


Author: R.J. Nijboer	Date of issue: 2005-Apr-18 Kind of issue: Public	Scope: Project Documentation Doc.nr.: LOFAR-ASTRON-DOC-???	
	Status: Draft Revision nr.: 0.1	File: ftp://rnijboer.astron.nl/???	


UVBrick

R.J. Nijboer

Verified:			
Name	Signature	Date	Rev.nr.
J.E. Noordam	o.p.v.	2005-??-??	0.1
J.E. Noordam	

Accepted:		
Work Package Manager	System Engineering Manager	Program Manager
J.E. Noordam	C.M. de Vos	J. Reitsma
.....
<i>date:</i>	<i>date:</i>	<i>date:</i>

©ASTRON 2005
All rights are reserved. Reproduction in whole or in part is prohibited without the written consent of the copyright owner.


Author: R.J. Nijboer	Date of issue: 2005-Apr-18 Kind of issue: Public	Scope: Project Documentation Doc.nr.: LOFAR-ASTRON-DOC-???	
	Status: Draft Revision nr.: 0.1	File: ftp://rnijboer.astron.nl/???	

Distribution list:

Group:	For Information:
ASTRON R. Assendorp M. Mevius J.E. Noordam T.A. Oosterloo O.M. Smirnov NRC A.G. Willis	ASTRON K. van der Schaaf C.M. de Vos


Document revision:

Revision	Date	Section	Page(s)	Modification
0.1	2005-Apr-18	-	-	Creation

Author: R.J. Nijboer	Date of issue: 2005-Apr-18 Kind of issue: Public	Scope: Project Documentation Doc.nr.: LOFAR-ASTRON-DOC-???	
	Status: Draft Revision nr.: 0.1	File: ftp://rnijboer.astron.nl/???	


Abstract

This document describes the UVBrick. The UVBrick is used for predicting *uv*-data of extended sources or of multiple point sources. In order to do so it takes a Sky Image (LSM Patch Image) and FFTs it into an image of *uv*-data. The UVBrick is implemented in the MeqTree system by two nodes: the MeqUVBrick, which contains an image of predicted *uv*-data, and the MeqUVInterpol, which fills a Request Cells / Vells using the data from the MeqUVBrick.

Author: R.J. Nijboer	Date of issue: 2005-Apr-18 Kind of issue: Public	Scope: Project Documentation Doc.nr.: LOFAR-ASTRON-DOC-???	
	Status: Draft Revision nr.: 0.1	File: ftp://rnijboer.astron.nl/???	

Contents

1	Introduction	5
2	'Predicting' sources via the UV-brick	5
3	Implementation of the UVBrick node	8
3.1	The Temporary Patch Image	8
3.2	Possible Improvements / Comments / etc.	9
4	Implementation of the UVInterpol node	9
4.1	Filling the Vells	10
4.2	The time / frequency to uv mapping	11
4.3	Interpolation	12
4.3.1	Bi-Cubic Hermite Interpolation	13
4.3.2	4th-Order Polynomial Interpolation	14
4.3.3	Bi-Linear Interpolation	14
4.4	Integration	14
4.5	Additional Results	14
4.6	Possible Improvements / Comments / etc.	14
5	An Example	15
6	To Do	15

Author: R.J. Nijboer	Date of issue: 2005-Apr-18 Kind of issue: Public	Scope: Project Documentation Doc.nr.: LOFAR-ASTRON-DOC-???	
	Status: Draft Revision nr.: 0.1	File: ftp://rnijboer.astron.nl/???	

1 Introduction

In the MeqTree system there is a need to predict uv -data for sources from the Local Sky Model (LSM). For single point sources this is done by means of a DFT Node. This node is connected to the LSM and directly fills a Request Cells / Vells with uv -data. For extended sources the prediction of uv -data is less trivial. It was decided that, for now, all extended sources will be dealt with using images. That is, an extended source (be it parameterized or an image of CLEAN components) will be described by means of a (Temporary) Patch Image (see document on the Local Sky Model, [1]). This image will be FFT'd into an uv -image. This uv -image can then be used to fill the Request Cells / Vells. In the MeqTree system this is implemented in two Nodes: the MeqUVBrick and the MeqUVInterpol. Multiple point sources can be treated in the same way.

The MeqUVBrick contains an image of uv -data. This image is contained in a Vells. At the moment we envisage the UVBrick to contain a 3D image (u , v , frequency) for one Stokes parameter. Every 4D Patch Image in the LSM will then be connected to four UVBrick's: one for each Stokes parameter.


Since the uv -data is stored in a Vells, the UVBricks can be added, subtracted, multiplied, etc. to form combinations. For instance, uv -data for correlation XX may be obtained by a MeqAdd node that has two UVBrick children: StokesI and StokesQ; see figure 1.

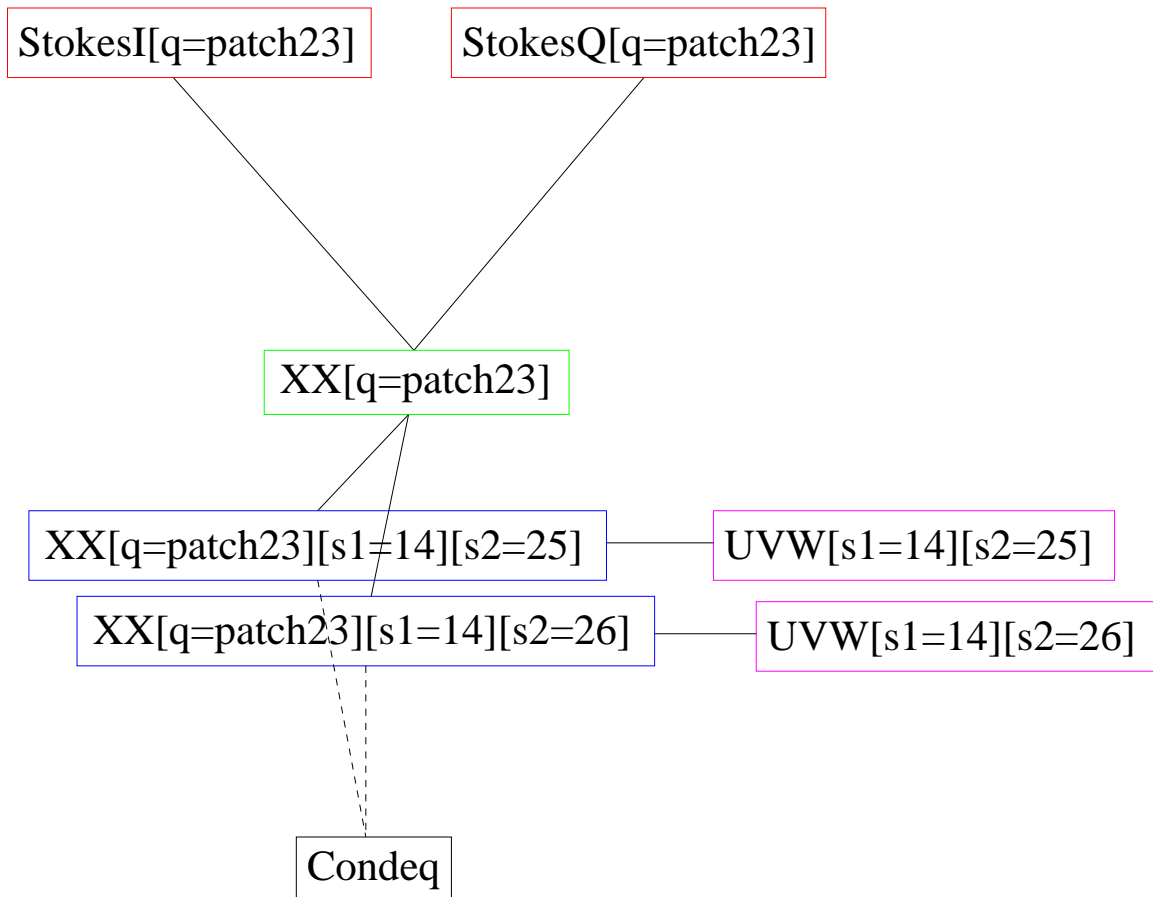
The node that contains the uv -data for one correlation, say XX, will then have a lot of parents: one MeqUVInterpol for each baseline. In figure 1 two such nodes are shown. The MeqUVInterpol will fill its Request Cells / Vells by integrating / interpolating uv -data from the uv -image of its child. Right now only the interpolation part is implemented, in a later stage it might prove necessary to also integrate over the Request Cell.

2 'Predicting' sources via the UV-brick

Some thoughts on the 'Predicting' of Extended Sources (adapted from the LSM document [1]):

- Filling the Temporary Patch Image is done in Glish / AIPS++. The MeqUVBrick node must then connect to the LSM and its Temporary Patch Image. The FFT into the UVbrick Image is done using AIPS++ FFT routines. The filling of the MeqDomain / Cells / Vells is ultimately done by the MeqUVInterpol node. The interface with the LSM must still be constructed, since no LSM is available yet.
- Source parameters are in Stokes I, Q, U, and V. So the Temporary Patch Image is in I, Q, U, and V. This image is FFT'd in the sky coordinates, yielding uv -data for I, Q, U, and V. Right now we envisage 4 3D UVBricks, each containing uv -data for one Stokes parameter. However, one 4D image containing uv -data for all Stokes parameters might also be an option. The Request Cells/ Vells want

Author: R.J. Nijboer	Date of issue: 2005-Apr-18 Kind of issue: Public	Scope: Project Documentation Doc.nr.: LOFAR-ASTRON-DOC-???	
	Status: Draft Revision nr.: 0.1	File: ftp://rnijboer.astron.nl/???	




MeqUVBrick

MeqAdd

MeqUVInterpol


MeqUVW

Figure 1: A Tree containing two MeqUVBricks and a number of MeqUVInterpols.

Author: R.J. Nijboer	Date of issue: 2005-Apr-18 Kind of issue: Public	Scope: Project Documentation Doc.nr.: LOFAR-ASTRON-DOC-???	
	Status: Draft Revision nr.: 0.1	File: ftp://rnijboer.astron.nl/???	

data for 1 correlation (say XX, XY, YX, or YY). Since the uv -data is stored in a Vells, we can use VellsMath to create the right correlation data.

- The FFT can be performed using the AIPS++ FFT engine (object: ImageFFT, function: fftsky) from the Images class. Make sure the right FFT is used (real vs. complex). The image data is always real. uv -data is complex. Beware of sign definitions.
- An image is defined as an Array of pixels, a Boolean Mask, and Coordinates. Therefore it seems slightly different from the Parameter Table for the MeqParm.
- The Temporary Patch Image and the UVbrick Image are implemented in AIPS++ as PagedImage (or Temporary Image??). In the MeqUVbrick node this AIPS++/C++ object can be used.
- There will be four UVbrick nodes: one for every Stokes parameter / Correlation. Each brick is thus a 3D dataset taking input from the same 4D UVbrick Image. These 4 UVbrick nodes will be available for the complete reduction. That is, as long as the number of frequency planes remains the same. Ideally, the number of frequency planes does not change often during a reduction.
- Still, the MeqUVbrick should be able to change the number of frequency planes automatically (new Temp Patch Image, new FFT).
- Also, since the UVBrick should be valid for all baselines, it should be valid for all values of w . For WSRT we only have $w = 0$, so this is no problem. For LOFAR $w \neq 0$ so we need to think of something.
- The number of frequency planes is determined by the Request.
- The UVImage must be constructed in such a way that the uv -data varies linearly over the grid points.
- The MeqDomain is filled by interpolation. At the moment one can choose from three different methods: 1) Bi-Cubic Hermite interpolation, where the image derivatives are determined numerically by central differences (default method); 2) 4th-order polynomial fit in 2D; 3) Bi-Linear interpolation.
- We should think about how to incorporate the effect of Cas. A. It is far away so this would yield rapidly varying uv data, hence fine grid spacings. This is probably to inefficient, so we might end up treating Cas. A with a DFT. (Does Cas. A cancel out automatically, right now?)
- At first we concentrate at the 'Predict' aspect of the MeqUVbrick. However, at some point we want to consider the aspect of 'Solving'. For this aspect perturbed values must be available.
- With the UVbrick there will be three ways to fill a MeqDomain with data: A predict Tree using the DFT Node and MeqParms, the MeqUVInterpol and the MeqUVbrick nodes, and from a Measurement Set using the MeqSpigot.

Author: R.J. Nijboer	Date of issue: 2005-Apr-18 Kind of issue: Public	Scope: Project Documentation Doc.nr.: LOFAR-ASTRON-DOC-???	
	Status: Draft Revision nr.: 0.1	File: ftp://rnijboer.astron.nl/???	

3 Implementation of the UVBrick node

The UVBrick node should be made only once (or a few times) during the reduction process and be valid for Requests with different time domains. This is possible whenever the frequency domain remains unchanged. Hence, Requests should have the same frequency domain / grid as much as possible.

Whenever the MeqUVBrick gets a Request it checks whether a uv -image exists and, when it does, whether the uv -image is valid for this Request. If so, the existing uv -image is used. Otherwise a new uv -image is constructed.

The resulting uv -image is stored in a Vells. One 3D uv -image per Plane. The uv -image has u , v , and f as coordinates. Hence, for the MeqUVBrick the Request Cells and the Result Cells do not match! This is ok, since only the frequency planes of the Request Cells are used to determine the frequency planes of the uv -image. (Note that a regular frequency grid is needed right now!)

3.1 The Temporary Patch Image

The Temporary Patch Image is part of the LSM. Since the LSM is not defined / implemented yet, this image is not yet available. Therefore, it is implemented in the UVBrick node by means of the function *makeUVImage*. In the future, this function must be replaced by a connection to the LSM.

The uv -image should have the same frequency planes as the Request Cells / Vells. In order to minimize the number of times a new uv -image must be made, the frequency planes of the Request must not change too often.

The uv -image is obtained by means of a FFT of an actual image. It reads


$$\vec{I}(u, v, w) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \mathbf{A}(l, m) \vec{I}(l, m) \exp^{-2\pi i[ul+vm+w(\sqrt{1-l^2-m^2}-1)]} \frac{dl dm}{\sqrt{1-l^2-m^2}}, \quad (1)$$

see equation (2-21) of [3]. For now, we consider $w = 0$, \mathbf{A} equals the identity matrix and we neglect the square root (which is approximately ok for small images: $l^2, m^2 \ll 1$). Note that by making \mathbf{A} dependent on station number and time and frequency we can apply image plane effects.

In equation (1) the u, v, w baseline coordinates are expressed in wavelengths. The AIPS++ FFT engine we are using is based on these wavelength coordinates. In the MeqTree system, however, baselines are expressed in meters. Hence, these coordinates must be transformed into one another as follows:

$$(u, v, w) = f (d_x, d_y, d_z) / c, \quad (2)$$

where f denotes the frequency and c the speed of light. d denotes the baseline length in meters. As a consequence the coordinates system will differ per frequency plane!

Author: R.J. Nijboer	Date of issue: 2005-Apr-18 Kind of issue: Public	Scope: Project Documentation Doc.nr.: LOFAR-ASTRON-DOC-???	
	Status: Draft Revision nr.: 0.1	File: ftp://rnijboer.astron.nl/???	

Right now, the MeqUVBrick node uses wavelength coordinates and the MeqUVInterpol transforms these into meters.

For the construction of the images we make the following observations:


- The maximum baseline (in wavelength) determines the grid spacing in the original image according to: $\Delta l = 1/(u_{max} - u_{min})$. (Note $u_{min} = -u_{max}$.)
- The size of the image and the grid spacing determine the number of pixels of the image. This is also the number of pixels in the uv -image.
- To be sure that all baselines fit into the uv -image we decrease the image grid spacing slightly. Let l_{max} be the maximum distance out of the phase center. Then the uv -grid spacing in u is given by $\Delta u = 1/(2l_{max})$. The grid spacing in RA is the given by $\Delta l = 1/(2(u_{max} + \Delta u)) = 1/(2u_{max} + 1/l_{max})$.
- We want to have a uv -image that varies linearly over the grid cells. In order to obtain this we demand 10 points per wavelength in the uv -image. This means that in our original image only the inner 1/10-th part contributes, or that the outer 9/10-th should be equal to zero. (Is a factor of $\sqrt{2}$ needed for diagonal propagation?)
- Images are stored as AIPS++ PagedImage objects.
- Right now all images are 4D. However, only the StokesI component is put into the Vells. This makes the Vells 3D.

3.2 Possible Improvements / Comments / etc.

- Vells as object in the UVBrick instead of the Paged Image. In this way the Vells are already there and do not need to be filled every time a new Request comes in.
- Use AIPS++ Temporary Image object instead of AIPS++ Paged Image object.
- Check AIPS++ FFT: # grid points, phase center, etc.
- Make construction of 'Curvature Maps' dependend on Request.
- UVImage coordinates in meters instead of wavelengths.

4 Implementation of the UVInterpol node

The MeqUVInterpol node fills a time/frequency Request Cells with uv -data from the MeqUVBrick node. Every frequency cell corresponds to a frequency plane of the uv -image. The temporal behaviour traces

Author: R.J. Nijboer	Date of issue: 2005-Apr-18 Kind of issue: Public	Scope: Project Documentation Doc.nr.: LOFAR-ASTRON-DOC-???	
	Status: Draft Revision nr.: 0.1	File: ftp://rnijboer.astron.nl/???	

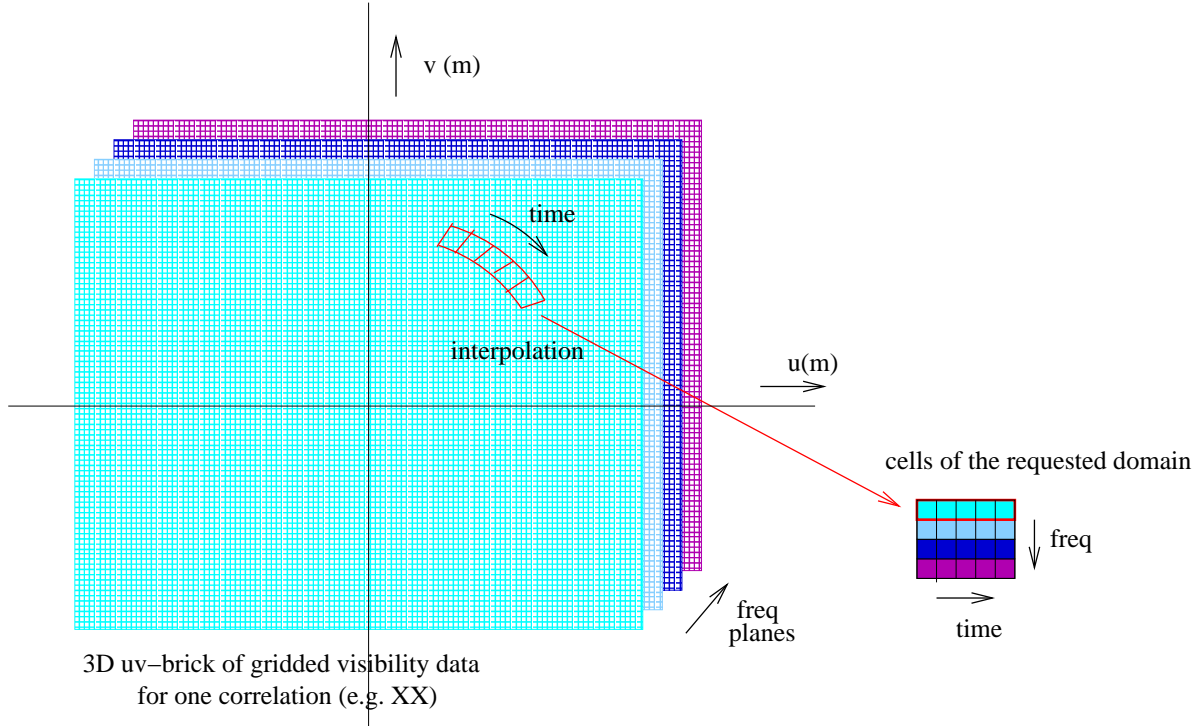


Figure 2: Integration / Interpolation from the 3D UVImage into the Request Cells as implemented in the MeqUVInterpol.


an elliptical band through the uv plane of a single frequency plane. This is depicted in figure 2. The fact that a single time / frequency Cell has an extension both in time and frequency transforms into a region in the uv plane. The extend of this region is determined by using info from a MeqUVW node.

4.1 Filling the Vells

For every cell of the Request Cells the value we have to find is

$$\vec{I}(t, f) = \frac{1}{\Delta t \Delta f} \int_{t-\Delta t/2}^{t+\Delta t/2} \int_{f-\Delta f/2}^{f+\Delta f/2} \vec{I}(u(\tilde{t}, \tilde{f}), v(\tilde{t}, \tilde{f})) d\tilde{t} d\tilde{f}. \quad (3)$$

Expression (3) indicates that in order to find $\vec{I}(t, f)$ we first have to determine $\vec{I}(u, v)$ and then integrate over the cell. The problem here is that a rectangular time / frequency cell maps onto an elliptical uv cell in the UVImage. The determination of $\vec{I}(u, v)$ is done in two steps: first find $u(t, f)$ and $v(t, f)$ and after that find an interpolated value for \vec{I} .

Author: R.J. Nijboer	Date of issue: 2005-Apr-18 Kind of issue: Public	Scope: Project Documentation Doc.nr.: LOFAR-ASTRON-DOC-???	
	Status: Draft Revision nr.: 0.1	File: ftp://rnijboer.astron.nl/???	

If the time-frequency cell is small, the variation of the UVImage over the cell is linear (by construction) and the center value is a (good?) approximation of the integrated cell value. For large time / frequency cells an additional integration scheme must be implemented.

4.2 The time / frequency to uv mapping

From the Request Cells we can determine for every time / frequency cell the central time / frequency value and the lower and upper bound of both the time range and the frequency range. Using the MeqUVW node we can find the central values for the corresponding d_x, d_y, d_z coordinates. Then, by assuming an elliptical time behaviour lower and upper bounds on the u, v coordinates can be determined.

For the elliptical curves the definition of Thompson, Moran, and Swenson [4], Chapter 4, is used:

$$\begin{aligned}
d_x(t) &= \sin(H(t))X + \cos(H(t))Y \\
d_y(t) &= \sin(\delta) (-\cos(H(t))X + \sin(H(t))Y) + \cos(\delta)Z \\
d_z(t) &= \cos(\delta) (\cos(H(t))X - \sin(H(t))Y) + \sin(\delta)Z
\end{aligned} \tag{4}$$

Note that

$$d_z(t) = -\frac{\cos(\delta)}{\sin(\delta)}d_y(t) + Z$$

As shown in [4] u and v lie on a elliptical curve in the uv -plane:


$$d_x^2(t) + \left(\frac{d_y(t) - Z \cos(\delta)}{\sin(\delta)} \right)^2 = X^2 + Y^2 \tag{5}$$

The change in u and v can be found to be:

$$\begin{aligned}
d_x(t + \Delta t) &= d_x(t) \cos(\Delta H) - \frac{d_y(t) - Z \cos(\delta)}{\sin(\delta)} \sin(\Delta H) \\
d_y(t + \Delta t) &= (d_y(t) - Z \cos(\delta)) \cos(\Delta H) + \sin(\delta) d_x(t) \sin(\Delta H) + Z \cos(\delta)
\end{aligned} \tag{6}$$

Note that these d_x, d_y, d_z coordinates are still in meters. Now, by using equation (2) these coordinates can be transformed into wavelengths. Using the central frequency value on the central d_x, d_y, d_z values yields a central u, v, w point. Using the lower and upper frequency bound on the lower and upper d_x, d_y, d_z , bounds yields four (when $w = 0$ or eight for $w \neq 0$) boundary points for the uv cell in the uv -image.

At the moment the values for δ and Z have to be set in the state record of the UVInterpol node. This can be done by setting the record fields `UVZ` and `UVDelta`. Note that when three time slots are available the values for δ and Z can also be found by parameter fitting on the central d_x, d_y, d_z values. However, this has not been implemented.

Author: R.J. Nijboer	Date of issue: 2005-Apr-18 Kind of issue: Public	Scope: Project Documentation Doc.nr.: LOFAR-ASTRON-DOC-???	
	Status: Draft Revision nr.: 0.1	File: ftp://rnijboer.astron.nl/???	

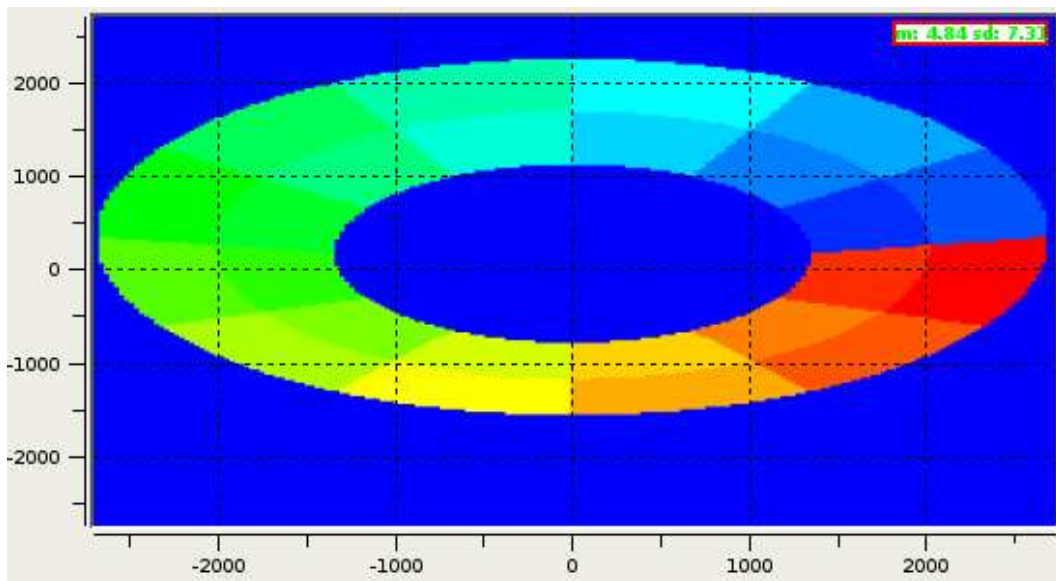


Figure 3: Mapping of a Cells onto the uv -domain.


In figure 3 a time / frequency domain for 24 hours and 150000000.0 Hz to 300000000.0 Hz having 12 cells in time and 2 in frequency is mapped onto the uv -plane. This was done for the values: $X = 0$, $Y = 2700$, $Z = 500$, and $\delta = \pi/4$. The uv domain ranged from -2700 up to + 2700 both in u and v . Note that all frequency planes are projected onto the same image. In practice the time cells corresponding to the same frequency cell are traced in the same frequency plane of the uv -image. However, different frequency cells correspond to different frequency planes in the uv -image.

4.3 Interpolation

For the Interpolation part we have so far implemented three methods:

1. Bi-Cubic Hermite Interpolation
2. 4th-order polynomial interpolation
3. Bi-linear Interpolation

In the MeqUVInterpol node one can choose the method by setting `Method` equal to 1, 2, or 3 respectively. Method 1 (Bi-Cubic Hermite interpolation) is the default method.

Author: R.J. Nijboer	Date of issue: 2005-Apr-18 Kind of issue: Public	Scope: Project Documentation Doc.nr.: LOFAR-ASTRON-DOC-???	
	Status: Draft Revision nr.: 0.1	File: ftp://rnijboer.astron.nl/???	

4.3.1 Bi-Cubic Hermite Interpolation

In one dimension Bi-Cubic Hermite interpolation is given by

$$p(x) = sf_1 + (1-s)f_0 + s^2(1-s) \left(-(x_1 - x_0)f'_1 + f_1 - f_0 \right) + s(1-s)^2 \left((x_1 - x_0)f'_0 - f_1 + f_0 \right), \quad (7)$$

where $x_0 \leq x \leq x_1$, $s = \frac{x-x_0}{x_1-x_0}$, $f_0 = f(x_0)$, $f_1 = f(x_1)$, $f'_0 = \frac{df}{dx}(x_0)$, and $f'_1 = \frac{df}{dx}(x_1)$. This is an approximation that is fourth order accurate in the grid spacing. However, both function values and values for the first derivatives are needed. The latter we will approximate numerically.

On a equidistant grid we can approximate the derivatives by a central difference, which is second order accurate in the grid spacing

$$f'_0 = \frac{f_1 - f_{-1}}{x_1 - x_{-1}}, \quad f'_1 = \frac{f_2 - f_0}{x_2 - x_0}. \quad (8)$$

This gives the following interpolation scheme

$$p(x) = sf_1 + (1-s)f_0 + s^2(1-s) \left(f_1 - \frac{1}{2}(f_2 + f_0) \right) + s(1-s)^2 \left(f_0 - \frac{1}{2}(f_1 + f_{-1}) \right). \quad (9)$$

Since the derivatives are approximated with second order accuracy, the resulting interpolation method remains fourth order accurate.

In order to extend the method to two dimensions, we first apply it in one direction (on four neighbouring grid-lines) and then in the second direction. We define

$$s = \frac{u - u_0}{u_1 - u_0}, \quad t = \frac{v - v_0}{v_1 - v_0}, \quad (10)$$


$$f_{u,ij} = \frac{1}{2} (f_{i+1,j} + f_{i-1,j}), \quad f_{v,ij} = \frac{1}{2} (f_{i,j+1} + f_{i,j-1}), \quad (11)$$

$$f_{uv,ij} = \frac{1}{4} (f_{i+1,j+1} + f_{i+1,j-1} + f_{i-1,j+1} + f_{i-1,j-1}).$$

The interpolation scheme then reads

$$\begin{aligned} p(u, v) = & tsf_{1,1} + t(1-s)f_{0,1} + (1-t)sf_{0,1} + (1-t)(1-s)f_{0,0} \\ & + ts^2(1-s)(f_{1,1} - f_{u,11}) + ts(1-s)^2(f_{0,1} - f_{u,01}) \\ & + (1-t)s^2(1-s)(f_{1,0} - f_{u,10}) + (1-t)s(1-s)^2(f_{0,0} - f_{u,00}) \\ & + t^2(1-t)s(f_{1,1} - f_{v,11}) + t^2(1-t)(1-s)(f_{0,1} - f_{v,01}) \\ & + t(1-t)^2s(f_{1,0} - f_{v,10}) + t(1-t)^2(1-s)(f_{0,0} - f_{v,00}) \\ & + t^2(1-t)s^2(1-s)(f_{1,1} - f_{u,11} - f_{v,11} + f_{uv,11}) \\ & + t^2(1-t)s(1-s)^2(f_{0,1} - f_{u,01} - f_{v,01} + f_{uv,01}) \\ & + t(1-t)^2s^2(1-s)(f_{1,0} - f_{u,10} - f_{v,10} + f_{uv,10}) \\ & + t(1-t)^2s(1-s)^2(f_{0,0} - f_{u,00} - f_{v,00} + f_{uv,00}). \end{aligned} \quad (12)$$

By pre-calculating f_u , f_v , and f_{uv} in the UVBrick, this interpolation scheme only uses table look up, which makes it relatively fast. Of course, at the expense of additional memory use in the UVBrick. The

Author: R.J. Nijboer	Date of issue: 2005-Apr-18 Kind of issue: Public	Scope: Project Documentation Doc.nr.: LOFAR-ASTRON-DOC-???	
	Status: Draft Revision nr.: 0.1	File: ftp://rnijboer.astron.nl/???	

'Curvature Map' images for f_u , f_v , and f_{uv} are stored in respectively plane 1, plane 2, and plane 3 of the UVBrick Result Vells. (Plane 0 contains the original image.)

Note 1: this method is probably related to the Bicubic Interpolation scheme of [2], section 3.6. The difference is that we pre-calculated the gradients.

Note 2: A Bicubic Spline interpolation seemed to yield only the same accuracy as Bi-Linear Interpolation. This seems strange or is it because the derivatives are determined globally? There also might still be an error in it. Bicubic splines are not implemented.

4.3.2 3rd-Order Polynomial Interpolation

The above described Bi-Cubic Hermite Interpolation scheme uses a 4×4 grid point stencil. Instead we can do a 2D polynomial fit on a 4×4 grid. This is implemented using the routines described in [2] section 3.6 (Higher Order for Accuracy). By using 4 grid points in both directions we use a third order polynomial fit in each direction. Note that a polynomial fit of order higher than 5 generally leads to inaccurate results.

This method is giving the same accuracy (or slightly better) as the Bi-Cubic Hermite interpolation. It is rather fast, although probably a bit slower than the Bi-Cubic Hermite interpolation. This should really be tested. On the other hand, the method does not need pre-computed values of the gradients. So it needs less memory than the Bi-Cubic Hermite interpolation.

4.3.3 Bi-Linear Interpolation

Bi-Linear interpolation is given by ([2], section 3.6)


$$p(u, v) = tsf_{1,1} + t(1-s)f_{0,1} + (1-t)sf_{0,1} + (1-t)(1-s)f_{0,0}. \quad (13)$$

Note that these are just the first terms of the Bi-Cubic Hermite interpolation.

Bi-Linear Interpolation is fast (it should be faster than both previous methods) and it does not need the gradients. However, it is only second order accurate, which is probably too inaccurate for our purposes.

4.4 Integration

For now the integration is just a simple evaluation of the center value. This should be approximately ok for small time / frequency cells, where there are zero or one uv grid points in the time / frequency cell. For larger cells an actual integration scheme must be implemented. (Possibly a double Romberg integration, see [2])

Author: R.J. Nijboer	Date of issue: 2005-Apr-18 Kind of issue: Public	Scope: Project Documentation Doc.nr.: LOFAR-ASTRON-DOC-???	
	Status: Draft Revision nr.: 0.1	File: ftp://rnijboer.astron.nl/???	

4.5 Additional Results

In the MeqUVBrick we built in the option of having additional results. By setting the state record field `Additional_Info` equal to `T` the MeqUVBrick result will also contain the records `uvinterp_count`, `uvinterp_map` and `uvinterp_image`. The result `uvinterp_count` contains the number of grid points from the uv image of the MeqUVBrick that fall into a cell. The result `uvinterp_map` shows which region of the uv -image is covered by the Request. A pixel that lies in cell (i, j) (where i is time and j is frequency) is given a value $j + nf * i + 1$ (nf denotes the number of frequency cells). The result `uvinterp_image` contains the first frequency plane of the UVImage (from the UVBrick) and of the three 'Curvature Maps'.

By setting the state record field `Additional_Info` equal to `F`, this additional info is not generated and the system is not burdened with it (Note that there will be a MeqUVInterpol node for every baseline. Putting in every one of them a complete image would consume a lot of memory.) By using the command `msg.setnodestate('uvinterp_node_name', [Additional_Info=T])` and re-executing, we can add the additional result to a specific node afterwards.


4.6 Possible Improvements / Comments / etc.

- Change order of time / freq. in double 'for' loop.
- Choose interpolation method ('if-then-else') outside double 'for' loop.
- Number of multiplications can be reduced in Bi-Cubic Hermite Interpolation scheme.
- Grid point search can be optimised.
- UVImage coordinates in meters instead of wavelengths.
- Finding values for Z and δ from fitting to UVW Node values.

5 An Example

6 To Do

- Check u, v, w coordinates (wavelengths vs. meters) in MeqUVBrick.
- $w \neq 0$.
- Can we take $\sqrt{1 - l^2 - m^2}$ fully into account?
- Can we take A into account?

Author: R.J. Nijboer	Date of issue: 2005-Apr-18 Kind of issue: Public	Scope: Project Documentation Doc.nr.: LOFAR-ASTRON-DOC-???	
	Status: Draft Revision nr.: 0.1	File: ftp://rnijboer.astron.nl/???	

- Can we have a non-regular frequency grid?

References

- [1] R.J. Nijboer, J.E. Noordam, *The Local Sky Model*, LOFAR document (in progress) 2005.
- [2] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, *Numerical Recipes, in C or Fortran, The Art of Scientific Computing*, Cambridge University Press, Cambridge, 1986 - 1992.
- [3] G.B. Taylor, C.L. Carilli, R.A. Perley, *Synthesis Imaging in Radio Astronomy II*, Astronomical Society of the Pacific Conference Series Volume 180, 1999.
- [4] A.R. Thompson, J.M. Moran, G.W. Swenson, Jr., *Interferometry and Synthesis in Radio Astronomy*, 2nd edition, John Wiley & Sons, Inc., New York, 2001.