MASSACHUSETTS INSTITUTE OF TECHNOLOGY
HAYSTACK OBSERVATORY
WESTFORD, MASSACHUSETTS 01886-1299

September 18, 2002

Phone: (978) 692-4764
Fax :   (781) 981-0590

To: LOFAR Group

From: RJ Cappallo

Subject: Visibility Integration

One step in the LOFAR simulator requires an integration in the UV plane of the Fourier-transformed sky brightness distribution. The domain over which the complex visibility function is integrated is a two-dimensional patch, with two of the edges following elliptical curves and two radial edges. The fractional width in the radial direction is equal to $\Delta f/f$, and the length of the elliptical boundaries is determined by the duration of the integration. The precise form of the elliptical boundary could be represented by (e.g.) semi-major axis, semi-minor axis, center location (along the V axis), and start and stop angles; it is determined by the geometry of the baseline relative to the center of the beam. As seen below, the approach I suggest eliminates the need to specify the shape of this elliptic boundary.

There are two distinct components to the complex visibilities, comprised of those sources that are within a defined primary beam, and those sources that lie outside of the beam. All sources in our sky model that lie within the beam contribute to a brightness distribution in the sky plane, which is Fourier transformed into an equivalent distribution in a region of the UV plane. Outside of the primary beam, only the bright sources are considered, and they are handled individually by the software. Each source is transformed via DFT to an analytic representation of that source's visibility contribution in the UV plane. For example, point sources become sinusoidal corrugations in the UV plane, and might be represented by two wave numbers and an amplitude. In general the bright Out of Beam Sources (OBS) will have visibilities that vary on a short spatial scale, while the undulations due to the in-beam sources (IBS) are more gradual.

I have investigated the analytic integration of the OBS visibilities over the elliptical patch, and found no easy way to perform the integration, even for the simple case of point sources. This is disappointing, since most of the contribution of these ripples will "integrate out" over the patch, and it is just the unmatched portions near the boundary that will determine the value of the integral. Some loss of accuracy becomes inevitable when integrating such a function numerically.

Since there appears to be no easy way to accomplish the UV integration analytically, I propose that we integrate numerically, in the FT (frequency-time) plane. The advantages of using the FT plane are that the domain becomes regular (a rectangle) and the natural weighting there is the same as what the correlator implicitly uses (see Figure 1). I see no disadvantage to using the FT plane, given that we are forced to numerically integrate. (If we were to integrate the OBS contribution analytically, in the UV plane the kernel might have a simple functional form, which is not the case when the corrugations are transformed to the FT plane.)

The integration in the FT plane must be performed with a relatively fine grid spacing, so that the rapid variations in the OBS component are adequately sampled. The IBS component is a two-dimensional grid of complex visibilities in the UV plane. A 16-point tabular interpolation function (see Appendix A) is used on the nearby IBS data, in order to generate values at the precise UV coordinates that correspond to the center of the FT cell. Each of the OBS components is evaluated functionally at the desired UV point, and added into the FT kernel.

Note that in the FT integration scheme there is no need for enormous IBS transforms; we need only sample the visibilities frequently enough (perhaps a 2k·2k UV grid will do) so that the interpolator will supply an accurate value.

The main computational burden of this approach lies in the small cell size in the FT plane, which will necessitate evaluation of the kernel a large number (~MN) of times. Each kernel evaluation entails:

- calculation of (u, v) at $(f_1+m\varDelta f, t_1+n\varDelta t)$

- interpolation of the IBS visibility grid at (u, v)

- evaluation (and summation) of the OBS visibility expression at (u, v) for *each* OBS source

The cell size in the FT plane is determined by the criterion that the integration error be kept at an acceptable level. The primary source of error will arise from the rapid undulations of the OBS visibility functions, since the low-order (cubic, for the algorithm in Appendix B) polynomial representation, which is implicit in the integrator, is inadequate to properly model the contribution of the higher order derivatives of the OBS functions. For example, if an out-of-beam point source is a factor of k times the (padded) field of view from the center of the FOV, then the phase of the sinusoidal corrugations will change by $k\pi$ radians across one UV grid cell.

Any standard two-dimensional quadrature algorithm may be employed for the FT integration. The algorithm we used may be found in Appendix C.
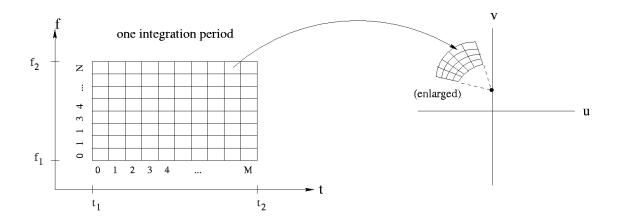
Figure 1. A single integration period in the FT plane is rectangular in shape, and corresponds to a domain in the UV plane having 2 elliptical and 2 linear boundaries. The numerical integration is performed over multiple cells, whose size of $\Delta f \times \Delta t$ is determined by the properties of the visibility function across the corresponding cell in the UV plane.

Appendix A – Interpolation

In order to represent the function being interpolated with sufficient accuracy, it was found necessary to use a grid of 16 nearby points, utilizing cubic polynomials along both axes. A straightforward extension of the Lagrange 4 point scheme given by Abramowitz & Stegun (eq. 25.2.13, p. 879) to two dimensions is given here.

First let us define the interpolating coefficients c_i (p) by:

$$c_{-1} = \frac{-p(p-1)(p-2)}{6}$$

$$c_0 = \frac{(p^2-1)(p-2)}{2}$$

$$c_1 = \frac{-p(p+1)(p-2)}{2}$$

$$c_2 = \frac{p(p^2-1)}{6}$$

Similar coefficients c_i (q) can also be defined, where p and q ($0 \leqq p,q \leqq 1$) will be the fraction of the distance between the central tabular points for the interpolation point. That is, $x = x_0 + p\Delta x$, and $y = y_0 + q\Delta y$. We then have the interpolated function values expressed as a linear combination of the surrounding 16 tabular points:

$$f(x,y) = \sum_{i=-1}^{2} \sum_{j=-1}^{2} c_i(p)c_j(q)f(x_i, y_j)$$

The error term in this expression is proportional to the 4th power of the cell spacing.
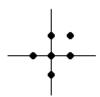
A C language prototype for an interpolating function using this algorithm could be written as:

double *sixteen_point* (double *grid*, int *m*, int *n*, double *p*, double *q*);

Where *grid* is the address of a 2-D array of data values, *m* & *n* are the indices into *grid* that identify the lower left corner of the interpolating cell, and *p* & *q* are both in the range (0..1), depicting the location of the interpolation point within the unit cell. Note that grid will have to have padding rows and columns, to ensure that *grid[m-1][n-1]* and *grid[m+2][n+2]* are both defined.

Appendix B – Alternative Interpolation Scheme

Originally, the function being interpolated was represented as a two dimensional 2nd order polynomial, by sampling at six grid points. We present it here in case it proves useful for efficiency reasons to return to this scheme, or use it elsewhere in the program. The chosen algorithm (Abramowitz & Stegun, equation 25.2.67, p. 882) uses the functional values at the grid points depicted below, in order to interpolate values in the unit cell whose lower-left corner is at the origin.

In this case, the interpolating function is

$$f(p,q) = \frac{q(q-1)}{2} f_{0,-1} + \frac{p(p-1)}{2} f_{-1,0}$$

$$+ (1 + pq - p^2 - q^2) f_{0,0} + \frac{p(p-2q+1)}{2} f_{1,0} + \frac{q(q-2p+1)}{2} f_{0,1} + pq f_{1,1}$$

where p, q are the x and y coordinates of the interpolation point, expressed as a fraction of the grid spacing. The values $f_{i,j}$ are the tabular function evaluated at grid point (i, j). The error term in this expression is proportional to the cube of the cell spacing.

A C language prototype for an interpolating function using this algorithm could be written as:

double *six_point* (double **grid*, int *m*, int *n*, double *p*, double *q*);

Where *grid* is the address of a 2-D array of data values, *m* & *n* are the indices into *grid* that identify the lower left corner of the interpolating cell, and *p* & *q* are both in the range (0..1), depicting the location of the interpolation point within the unit cell. Note that grid will have to have a padding row and column, to ensure that *grid*[*m*-1][*n*] and *grid*[*m*][*n*-1] are both defined.

Appendix C – Integration

Let us assume that the FT plane integration is being carried out from time t_1 to time t_2, and over the frequency band from f_1 to f_2. Furthermore, we will divide the time into M blocks, each of width $\Delta t = (t_2 - t_1) / M$, and the frequency into N intervals, each of width $\Delta f = (f_2 - f_1) / N$. We will use a quadrature formula based upon Simpson's rule extended into two dimensions (see Abramowitz & Stegun, eq. 25.4.62, p.892). Let $V(f, t)$ be the complex visibility at frequency f and time t. Then our integral can be approximated by the sum:

$$\int_{t_1}^{t_2}\int_{f_1}^{f_2} V(f,t)\,df\,dt \cong \frac{1}{9}\Delta f\,\Delta t \sum_{m=0}^{M}\sum_{n=0}^{N} V(f_1 + m\Delta f, t_1 + n\Delta t)\cdot W_{mn}$$

where the weighting matrix W_{mn} has the form:

$$W_{mn} = \begin{bmatrix} 1 & 4 & 2 & 4 & 2 & \dots & 4 & 1 \\ 4 & 16 & 8 & 16 & 8 & \dots & 8 & 4 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 2 & 8 & 4 & 8 & 4 & \dots & 8 & 2 \\ 4 & 16 & 8 & 16 & 8 & \dots & 16 & 4 \\ 2 & 8 & 4 & 8 & 4 & \dots & 8 & 2 \\ 4 & 16 & 8 & 16 & 8 & \dots & 16 & 4 \\ 1 & 4 & 2 & 4 & 2 & \dots & 4 & 1 \end{bmatrix}$$

The local error is of order $(\Delta f\,\Delta t)^2$, so the global error will scale as $(\Delta f\,\Delta t)$. For easy construction of W_{mn}, note that $W_{mn} = W_{m0}W_{0n}$. This permits the edge rows and columns to be easily filled in first, as they have a simple form, and then the rest of the matrix can be derived from them.