Chapter 15

The Imaging Principles

Stéphane Guilloteau

guillote@iram.fr

IRAM, 300 rue de la Piscine, F-38406 Saint Martin d'Hères, France

Assuming identical antennas, we have shown in previous lectures that an interferometer measures the *visibility function*

$$V(u,v) = \iint B(x,y)\, I(x,y)\, e^{-2i\pi(ux+vy)}\, dx\, dy \tag{15.1}$$

over an ensemble of points $(u_i, v_i), i = 1, n$, where $B(x,y)$ is the power pattern of the antennas and $I(x,y)$ the sky brightness distribution.

The imaging process consists in determining as best as possible the sky brightness $I(x,y)$. Since Eq.15.1 is a convolution, the imaging process will involve deconvolution techniques.

Let $S(u,v)$ be the sampling (or spectral sensitivity) function

$$\begin{aligned} &S(u,v) \neq 0 \Longleftrightarrow \exists i \in 1,n\,(u_i,v_i) = (u,v) \\ &S(u,v) = 0 \Longleftrightarrow \forall i \in 1,n\,(u_i,v_i) \neq (u,v) \end{aligned} \tag{15.2}$$

The spectral sensitivity function S contains information on the relative weights of each visibility, usually derived from noise predicted from the system temperature, antenna efficiency, integration time and bandwidth.

Let us define

$$I_w(x,y) = \iint S(u,v)\, W(u,v)\, V(u,v)\, e^{2i\pi(ux+vy)}\, du\, dv \tag{15.3}$$

where $W(u,v)$ is an arbitrary *weighting function*. Since the Fourier Transform of a product of two functions is the convolution of the Fourier Transforms of the functions, $I_w(x,y)$ can be identified with

$$I_w(x,y) = (B(x,y)I(x,y)) **(D_w(x,y)) \tag{15.4}$$

where

$$D_w(x,y) = \iint S(u,v)\, W(u,v)\, e^{2i\pi(ux+vy)}\, du\, dv = \widehat{SW} \tag{15.5}$$

$D_w(x,y)$ is called the *dirty beam*, and is directly dependent on the choice of the *weighting function* W, as well as on the spectral sensitivity function S. $I_w(x,y)$ is usually called the dirty image.

Fourier Transform, which allows to directly derive I_w from the measured visibilities V and spectral sensitivity function S, and **Deconvolution**, which allows to derive the sky brightness I from I_w, are thus two key issues in imaging (see Eq.15.4).

15.1 Fourier Transform

The first step in imaging is to evaluate the dirty image, using Fourier Transform. Several techniques are available.

15.1.1 Direct Fourier Transform

The simplest approach would be to directly compute sin and cos functions in Eq.15.4 for all combinations of visibilities and pixels in the image. This is straightforward, but slow. For typical data set from the VLA, which contain up to 10^5 visibilities per hour and usually require large images (1024×1024 pixels), the computation time can be prohibitive. On the other hand, the IRAM Plateau de Bure interferometer produces about 10^4 visibilities per synthesis, and only require small images (128×128). The Direct Fourier Transform approach could actually be efficient on vector computers for spectral line data from Plateau de Bure interferometer, because the sin and cos functions needs to be evaluated only once for all channels. Moreover, the method is well suited to real-time display, since the dirty image can be easily updated for each new visibility.

15.1.2 Fast Fourier Transform

In practice, everybody uses the Fast Fourier Transform because of its definite speed advantage. The drawback of the methods is the need to regrid the visibilities (which are measured at arbitrary points in the (u,v) plane) on a regular grid to be able to perform a 2-D FFT. This gridding process will introduce some distortion in the dirty image and dirty beams, which should be corrected afterwards. Moreover, the gridded visibilities are sampled on a finite ensemble. As discussed in more details below, this sampling introduces aliasing of the dirty image (and beam) in the map plane.

15.1.3 Gridding Process

The goal of gridding is to resample the visibilities on a regular grid for subsequent use of the FFT. At each grid point, gridding involves some sort of interpolation from the values of the nearest visibilities. The visibilities being affected by noise, the interpolating function needs not fit exactly the original data points. Although any interpolation scheme could a priori be used, such as smoothing spline functions, it is customary to use a *convolution* technique to perform the gridding.

Using a convolution is justified by several arguments. First, from Eq.15.1, $V = \widehat{BI} = \hat{B} * * \hat{I}$. Hence V is already a convolution of a (nearly Gaussian) function $\hat{B}$ with the Fourier Transform of I. Nearby visibilities are not independent. Second, as mentioned above, exact interpolation is not desirable, since original data points are *noisy samples of a smooth function.* Third, if the width of the convolution kernel used in gridding is small compared to $\hat{B}$, the convolution added in the gridding process will not significantly degrade the information. Last, but not least, it is actually possible to correct for the effects of the convolution gridding.

To demonstrate that, let G be the gridding convolution kernel. Eq.15.3 becomes

$$I_w^g \rightleftharpoons G * * (S \times W \times V) \tag{15.6}$$

We thus have for the image I the following relations

$$I^g_w = \hat{G} \times (\widehat{SW} **\hat{V}) = \hat{G} \times I_w \tag{15.7}$$

and for the dirty beams

$$D^g_w \rightleftharpoons G **(S \times W) \quad D^g_w = \hat{G} \times \widehat{SW} \tag{15.8}$$

from which we derive the relation

$$\frac{I^g_w}{\hat{G}} = \frac{D^g_w}{\hat{G}} **(BI) \tag{15.9}$$

Thus the dirty image and dirty beams are obtained by dividing the Fourier Transform of the gridded data by the Fourier Transform of the gridding function.

15.2 Sampling & Aliasing

Sampling on a regular grid is equivalent to multiplying by a series of periodically spaced delta functions, i.e. the so-called *shah* function III:

$$[\frac{1}{\Delta u}]\text{III}(\frac{u}{\Delta u}) = \sum_{k=-\infty}^{\infty} \delta(u - k\Delta u) \tag{15.10}$$

The Fourier Transform of the *shah* function above is the *shah* function

$$\text{III}(x\Delta u) = \frac{1}{\Delta u} \sum_{m=-\infty}^{\infty} \delta(x - \frac{m}{\Delta u}) \tag{15.11}$$

Hence, sampling the visibilities V results in convolving its Fourier Transform $\hat{V}$ by a periodic *shah* function. This convolution reproduces in a periodic way the Fourier Transform of the visibilities $\hat{V}$.

If the Fourier Transform of the visibilities $\hat{V}$, i.e. the brightness distribution BI, has finite support ΔX, the replication poses no problem provided the support is smaller than the periodicity of the *shah* function, i.e.

$$(\Delta u)^{-1} \geq (\Delta X) \qquad \Delta u \leq (\Delta X)^{-1} \tag{15.12}$$

If not, data outside $(\Delta u)^{-1}$ are *aliased* in the *imaged area* $(\Delta u)^{-1}$.

In aperture synthesis, finite support is ensured to first order by the finite width of the antenna primary beam B. However, strong sources in the antenna sidelobes may be aliased if the imaged area is too small. Moreover, the **noise** does not have finite support. White noise in the uv plane would result in white noise in the map plane. In practice, the noise in the uv plane is not completely white. However, it is support limited (since only a finite region of the uv plane is sampled in any experiment). Accordingly, its Fourier Transform in the map plane is not support limited. Noise aliasing thus occurs, and produces an increased noise level at the map edges.

15.3 Convolution and Aliasing

The combination of Gridding and Sampling produces the uv data set

$$V_m = \frac{1}{\Delta u \Delta v}\text{III}(\frac{u}{\Delta u}, \frac{v}{\Delta v}) \times (G **(S \times W \times V))(u, v) \tag{15.13}$$

$$= \text{III} \times (G **(S \times W \times V))/(\Delta u \Delta v) \tag{15.14}$$

which analogous with Eq.15.6

The Fourier Transform of this uv data set is

$$\widehat{V_m} = \mathrm{III}(x\Delta u, y\Delta v) **(\hat{G} \times (\widehat{SW} **\hat{V})) \quad (15.15)$$

$$= \mathrm{III} **(\hat{G} \times (\widehat{SW} **(BI))) \quad (15.16)$$

$\widehat{V_m}$ is thus the sky brightness multiplied by the primary beam (BI), convolved by the the dirty beam $\widehat{SW}$, then multiplied by the Fourier transform of the gridding function $\hat{G}$ and periodically replicated (by the convolution with the Shah function).

Accordingly, aliasing of $\hat{G}$ in the map domain will thus occur. Note at this stage that, providing aliasing of $\hat{G}$ remains negligible, an exact convolution equation is preserved

$$\frac{\widehat{V_m}}{\hat{G}} = \widehat{SW} **BI \quad (15.17)$$

The gridding function will thus have to be selected to minimize aliasing of $\hat{G}$. This criterion will depend on the image fidelity required. Obviously, if the data is very noisy, aliasing of the $\hat{G}$ can be completely negligible.

Furthermore, the weighting function W is usually smooth, while the gridding function G is a relatively sharp function (since it ensures the re-gridding by convolution from *nearby* data points). Thus, to first order $G **W = W$, and we could rewrite Eq.15.14 as

$$V_m = \mathrm{III} \times W \times (G **(S \times V))/(\Delta u \Delta v) \quad (15.18)$$

Hence, the weighting can be performed *after* the gridding. The choice of weighting before or after gridding is essentially based on computational speed or algorithmic simplicity.

Let us focus on the choice of the gridding function. The gridding function will be selected according to the following principles:

1. small support, typically one or two cells wide (Δu).
2. small aliasing.
3. fast computation.

Points 1 and 2 are contradictory, since a small support for G implies a large extent of $\hat{G}$. Some compromise is required. For simplicity, gridding functions are usually selected among those with separable variables:

$$G(u,v) = G_1(u)G_1(v)$$

although this could break the rotation invariance.

The simplest gridding function is the **Rectangular function**

$$G(u) = \frac{1}{\Delta u}\Pi(\frac{u}{\Delta u}) \quad (15.19)$$

$$\hat{G}(x) = \frac{sin(\pi \Delta u x)}{\pi \Delta u x} \quad (15.20)$$

where Π is the unit rectangle function. Obviously, aliasing will be important, since the `sinc` function falls off very slowly.

A better choice could be the **Gaussian function**

$$G(u) = \frac{1}{\alpha \Delta u \sqrt{\pi}} e^{-(u/\alpha\Delta u)^2} \quad (15.21)$$

$$\hat{G}(x) = e^{-(\pi \alpha x \Delta u)^2} \quad (15.22)$$

By proper selection of α (not too small, not too large), a compromise between computation speed (better for small α) and aliasing (better for large α) can be found. $\alpha = 2\sqrt{ln(4)} \simeq 0.750$ is a standard choice.

However, a Gaussian still has fairly significant wings. $\hat{G}$ should ideally be a rectangular function (1 inside the map, 0 outside). G would be a `sinc` function, but this falls off too slowly, and would require a

lot of computations in the gridding. Moreover, the (unavoidable) truncation of G would destroy the sharp edges of $\hat{G}$ anyhow. Hence the idea to use an apodized version of the `sinc` function, the **Gaussian-Sinc function**

$$G(u) = \frac{\sin \pi u/(\alpha \Delta u)}{\pi u} e^{-(u/(\beta \Delta u))^2} \tag{15.23}$$

$$\hat{G}(x) = \Pi(\alpha x \Delta u) * (\sqrt{\pi}\beta \Delta u e^{-(\pi \beta x \Delta u)^2}) \tag{15.24}$$

It provides good performance for $\alpha = 1.55$ and $\beta = 2.52$.

The empirical approaches mentioned above do not guarantee any optimal choice of the gridding function. A completely different approach is based on the desired properties of the gridding function. We actually want $\hat{G}$ to fall off as quickly as possible, but G to be support limited. Mathematically, this defines a class of functions known as **Spheroidal functions**. Spheroidal functions are solutions of differential equations, and cannot be expressed analytically. In practice, this is not a severe limitation since numerical representations can be obtained by tabulating the gridding function values. Given the limited numerical accuracy of the computations, the tabulation does not require a prohibitively fine sampling of the gridding function, and is quite practical both in term of memory usage and computation speed. Tabulated values are used in the task `UV_MAP`.

Note that the finite accuracy of the computation may ultimately limit the image dynamic range.

15.4 Error Analysis

We thus succeeded to preserve a convolution equation, with the slight restrictions due to the aliasing and gridding correction. Let us explore now what *errors* or *systematic* effects may appear in the image plane.

First, consider the **noise**. Aliasing increases the noise level at the map edges (by noise aliasing and then by the gridding correction since this amounts to divide by the Fourier Transform of the gridding function, which is unity at the map center, but smaller at the map edges). For example, the noise increases by a factor $(\pi/2)^2$ at map corners for the Gaussian-Sinc function. Near the map center, the effect is negligible. Note that for a given field of view, the noise increase can be arbitrarily limited by making a sufficiently large image, but this has a high computational price.

Concerning errors, it is important to separate two main classes of errors.

Additive errors

The Fourier transform being linear, additive errors result in artificial structure *added* to the true map, e.g.
- A single spurious visibility will produce fringes in the map
- An additive real term (correlator offset), will produce a point source at the phase tracking center.

Multiplicative errors

A multiplicative term on the visibility distorts the image, since

$$V(u,v) \times \epsilon(u,v) \longleftrightarrow \hat{V}(x,y) ** \hat{\epsilon}(x,y)$$

i.e. the map is convolved by the Fourier transform of the error. Calibration errors (in phase or amplitude) are of this type. Among these, the **seeing** should not be neglected.

Phase calibration errors result in antisymmetric patterns.

15.5 Weighing and Tapering

There is still a free parameter in the image construction process: the weighting function. At uv table creation, the sampling function is defined as

$$S(u,v) = \frac{1}{\sigma^2(u,v)} \tag{15.25}$$

where the noise σ is computed from system temperature, bandwidth, integration time, and system efficiency (including quantization and decorrelation).

$$\sigma(u,v) = \frac{\mathcal{J}_I T_{\rm sys}}{\eta_{\rm Q}\sqrt{2\Delta\nu t_{\rm int}}} \tag{15.26}$$

$\mathcal{J}_I$ being the antenna temperature to flux density conversion factor:

$$\mathcal{J}_I = \frac{2k}{\eta_A A} \tag{15.27}$$

The weights $W(u,v)$ can be freely chosen. The selecting of weights is usually decomposed in two slightly different processes, called Weighting and Tapering.

Weighting deals with the local variations of weights for each grid cell after the gridding process. Since the original uv coverage is an ensemble of ellipses, the gridding may leave a weight distribution with very large dispersion. Weighting can be applied to uniformize this distribution.

On the other hand, **Tapering** consists in apodizing the uv coverage by $T(u,v) = \exp(-(u^2+v^2)/t^2))$ where t is a tapering distance. This corresponds to smoothing the data in the map plane (by convolution with a gaussian).

The simplest possibility, called **Natural Weighting, without taper** is to keep the original spectral sensitivity function by setting $W(u,v) = 1$. This can be demonstrated to maximize sensitivity to *point sources* (i.e. sources smaller than the synthesized beam). Proper design (and use) of the array can ensure that the resulting synthesized beam is appropriate, in terms of size (angular resolution matched to the scientific goal) and shape (lowest possible sidelobes).

If the sources of interest are somewhat extended, tapering can be used to increase **brightness** sensitivity. Tapering may also have the advantage of lowering the sidelobes. This is usually true for limited tapering, which reduces the effect of the discontinuity at the outer edge of the uv plane, but is not the case for strong tapering, where the result becomes critically dependent on the actual sampling of the inner part of the uv plane. However, tapering is always throwing out some information, namely the long baselines part of the data set. Hence, it should be used either with moderate tapers, or as a complementary view on a data set. To increase brightness sensitivity, one should use preferentially compact arrays rather than tapering.

Uniform Weighting consists in selecting the weights $W(u,v)$ so that the sum of weights $\sum W \times S$ over a uv cell is a constant function (or zero if no uv data exists in that cell). The size (radius) of the uv cell is an arbitrary parameter. It can be the cell size resulting from the gridding process, i.e. the inverse of the field of view, but any other choice is possible. Using half of the dish diameter is well justified, since the visibilities are convolution of Fourier transform of the sky brightness by the Fourier transform of the primary beam. **Uniform Weighting** gives more weight to long baselines than natural weighting (because you spend less time per uv cell on long baselines than on short baselines for earth synthesis). **Uniform Weighting** produces smaller beam. Because it fills the uv plane more regularly, Uniform weighting could be thought also to produce lower sidelobes. However, because of the discontinuity of the weights at the edge of the sampled portion of the uv plane, the inner sidelobes tend to be increased, unless some tapering is combined with Uniform weighting.

Robust Weighting is a variant of uniform weighting which avoids to give too much weight to a uv cell with low natural weight. There are several ways to implement such a scheme. Roughly speaking, if the sum of natural weights in a cell is less than a threshold, the weighting is unchanged, if it is more, the weight is set to this threshold. Let S_n be the natural weight of a cell, and S_t a threshold for such weight. Robust weighting could be implemented by selecting the weight W as

$$\begin{aligned} S_n < S_t &\rightleftharpoons W = 1 \\ S_n > S_t &\rightleftharpoons W = S_t/S_n \end{aligned} \tag{15.28}$$

or a more continuous formula like

$$W = \frac{1}{\sqrt{1 + S_n^2/S_t^2}} \tag{15.29}$$

Robust weighting combines the advantages of **Natural** and **Uniform** weighting, by increasing the resolution and lowering the sidelobes without degrading too much the sensitivity. By adjusting the threshold, it approaches either case (large threshold $\longleftrightarrow$ Natural, small threshold $\longleftrightarrow$ Uniform).

Weighting and Tapering reduce point source sensitivity by

$$\sqrt{\sum T^2 W^2}/(\sum TW) \tag{15.30}$$

15.6 The GILDAS implementation

We have now introduced the basic parameters of the imaging process: gridding, weighting and tapering. The main imaging task in the GILDAS software is `UV_MAP`. Before using `UV_MAP`, it is also recommended to use the associated task `UV_STAT` which evaluates the beam sizes, point source and brightness sensitivity as function of taper or robust weighting parameter.

Although the choice of configurations for the Plateau de Bure interferometer has been performed in order to optimize the uv coverage for most observing conditions, robust weighting can often offer a better compromise, unless signal to noise is insufficient. Task `UV_STAT` also suggests appropriate pixel sizes for `UV_MAP`

The imaging task `UV_MAP` is controlled by the following parameters:

- `MAP_SIZE`
  The number of pixels in each direction. This should be powers of 2.
- `MAP_CELL`
  The pixel size, in arcsecond, in each direction. It should respect proper sampling compared to the synthesized beam width. In practice, $3-4$ pixels per beam width are required. Task `UV_STAT` can compute the optimum value for this parameter. Note that the imaged area is `MAP_SIZE` $\times$ `MAP_CELL`
- `MCOL`
  For spectral line data, the first and last channel to be imaged. (0,0) means all data.
- `WCOL`
  The channel from which the natural weights S are taken. `UV_MAP` produces only one beam for all channels (by default, there is an alternate option for experts). `WCOL = 0` is equivalent to `WCOL = (MCOL[1]+MCOL[2])/2`.
- `WEIGHT_MODE`
  `UN` for **Uniform** or `NA` for **Natural** weighting. `UN`iform weighting is actually a **Robust** weighting in `UV_MAP`.
- `UV_CELL`
  When UNiform weighting is used, `UV_CELL[1]` is the UV cell diameter (in meters), and `UV_CELL[2]` is the threshold for robust weighting: 1 corresponds to the mean natural weight of all cells. `UV_CELL[1]` should normally be 7.5 m for Plateau de Bure data.
- `CONVOLUTION`
  This is the convolution type for gridding. Choices are offered for test purposes, but `CONVOLUTION = 5` (Spheroidal) gives best results.

The other parameters are used to re-center the map (by phase shifting the uv data before imaging) when needed. This is convenient for **Mosaics**. `UV_MAP` performs all the imaging steps presented before: gridding, weighting, tapering, correction for gridding function, and computes the dirty beam and dirty image.

Both `UV_STAT` and `UV_MAP` are implemented as commands in the MAPPING program, or as tasks available from the GRAPHIC program. Using one or the other is a matter of personal preference.

15.7 Deconvolution

The first imaging step presented before leads to a convolution equation whose solution is the convolution product of the sky brightness distribution (apodized by the interferometer primary beam) by the dirty beam.

To derive the astronomically meaningful result, i.e. ideally the sky brightness, a deconvolution is required. Deconvolution is always a non linear process, and requires (in one way or another) to impose some constraints on the solution, or in other words to add some information, to better select plausible solutions. Such additional constraints can be explicit (e.g. positivity, or user specified finite support) or qualitative.

15.7.1 The CLEAN method

The standard deconvolution technique, CLEAN relies on such a qualitative constraint: it assumes that the sky brightness is essentially a ensemble of point sources (the sky is dark, but full of stars). The algorithm which derives from such an assumption is straightforward. It is a simple "matching pursuit"

1. Initialize a *Residual* map to the *Dirty* map
2. Initialize a *Clean component list* to zero.
3. Assume strongest feature in *Residual* map originates from a point source
4. Add a fraction γ (the *Loop Gain*) of this point source to the *Clean component list*, remove the same fraction, convolved with the dirty beam, from the *Residual* map.
5. If the strongest feature in the *Residual* map is larger than some threshold, go back to point 3 (each such step is called an iteration).
6. If the strongest feature is below threshold, of if the number of iterations $N_{\rm iter}$ is too large, go to point 7.
7. Convolve the *Clean component list* by a properly chosen *Clean Beam* (this is called the restoration step).
8. add to the result the *Residual* map to obtain the *Clean Map*.

The CLEAN algorithm as a number of free parameters. The loop gain controls the convergence of the method. In theory, $0 < \gamma < 2$, but in practice one should use $\gamma \simeq 0.1 - 0.2$, depending on sidelobe levels, source structure and dynamic range. While high values of γ would in principle give faster convergence, since the remaining flux is $\propto (1-\gamma)^{N_{\rm iter}}$ if the object is made of a single point source, deviations from an ideal convolution equation force to use significantly lower values in order to avoid non linear amplifications of errors. Such deviations from the ideal convolution equation are unavoidable because of thermal noise, and also of phase and amplitude errors which distort the dirty beam.

The threshold for convergence and number of iterations define to which accuracy the deconvolution proceeds. It is common practice to CLEAN down to about the noise level or slightly below. However, in case of strong sources, the residuals may be dominated by dynamic range limitations rather than by noise.

The clean beam used in the restoration step plays an important role. It is usually selected as a 2-D Gaussian, which allows the convolution to be computed by a simple Fourier transform, although other choices could be possible. The size of the clean beam is a key parameter. It should be selected to match the (inner part of) the dirty beam, otherwise the flux density estimates may be incorrect. To understand this problem, let us note first that the units of the dirty image are undefined. Simply, a 1 Jy isolated point source appears with a peak value of 1 in the dirty map. This is no longer true (because of sidelobes) if there is more than one point source, or a fortiori, an extended source. The unit of the clean image is well defined: it is Jy per beam, which can easily be converted to brightness temperature from the effective clean beam solid angle and the observing wavelength. Now, assume the source being observed is just composed of 2 separate point sources of equal flux, and that the dirty beam is essentially a Gaussian. Let us clean the dirty image in such a way that only 1 of the 2 point sources is actually included in the clean component list. If we restore the clean image with a clean beam which is, e.g. twice smaller than the original dirty beam, the final result will undoubtedly be odd. The second source would appear extended and have a larger flux than the first one. No such problem appears if the clean beam matches the dirty beam. Admittedly, the above example shows a problem which results from a combination of two effects: an inappropriate choice for the clean beam, and an insufficient deconvolution. However, the second problem always exists to some extent, because of noise in the original data set. Hence, to minimize errors, it is important to match the clean and dirty beams.

Note that in some circumstances, there may be no proper choice. An example is a dirty beam with narrow central peak on top of a broad "shoulder". Small scale structures will be properly reconstructed, but larger ones not.

The last step in the CLEAN method plays a double role. On one hand, it protects against insufficient deconvolution. Furthermore, since the residual image should be essentially noise if the deconvolution has converged, it allows noise estimate on the cleaned image.

15.7.2 Interpretation of CLEAN

If CLEAN converged, the *Clean component list* is a plausible solution of the measurement equation (within the noise), but it is not unique... Hence, because of convolution by the clean beam, the clean image is **not** a solution. However, besides allowing a reasonable definition of the image unit in case of incomplete convergence, there are two reasons to convolve by a clean beam. First, convolution by the clean beam smears out artifacts due to extrapolation beyond the measured area of the *uv* plane. This is an **a posteriori regularization**. Second, the clean components are forced to reside on the grid defined by the image. This discrete representation has a number of limitations (e.g. necessity of negative clean components, limited accuracy due to the finite size of the component list), which are reduced by convolution by the clean beam, because the clean image then has finite resolution and can be properly represented on a discrete grid provided the Nyquist sampling is preserved.

An important property of CLEAN is that (to first order) only the inner quarter of the dirty image can be properly cleaned. This is easily understood when dirty beam and dirty images are computed on the same grid size, since a source at one edge of the inner quarter requires knowledge of the dirty beam sidelobes beyond the map size to be deconvolved from the opposite edge. However, this also remains true if one computes the dirty beam on a twice larger grid than the dirty image: more than the inner quarter can be deconvolved, but because of aliasing, the map edges can never be.

Finally, CLEAN offers a very simple way to impose further constraints on the class of solution which is acceptable, by allowing definition of a *support*. This can be the standard (simple or multiple) *Clean Box* available in many non interactive implementations, or a user defined mask in interactive implementations. The search region can even be modified from iteration to iteration to help clean convergence. Such a flexible support is available inside the MAPPING program. Note however that the Clean Box or support should not be too limited: cleaning the noise is necessary too (as well as incorporating negative Clean component).

15.7.3 The CLEAN variants

The original CLEAN method is due to [Hogbom 1974]. Several variants exist.

One of the most popular (CLARK) is due to [Clark 1980], and involves minor and major cycles. In **Minor** cycles, an Hogbom CLEAN is performed, but with a truncated dirty beam, and only on the list of brightest pixels. This search is fast, because of the dirty beam truncation and because of the limited support. The Clean components identified during the minor cycles are removed at once by a FFT during a **Major** cycle. Because removal is done by FFT, slightly more than the inner map quarter can be cleaned.

A second variant, called MX, due to [Cotton & Schwab 1984], is similar to the CLARK method, except that the Clean components are removed from the *uv* table at the **Major** cycle stage (and thus the imaging process is repeated at each major cycle). This avoid aliasing of sidelobes, allows to clean more than the inner quarter, but is relatively slow because of the re-imaging at major cycles. Unless disk storage is a real problem, a faster result of equal quality is obtained by standard Clean with a twice larger map.

The next variant, called SDI (from [Steer et al. 1984]), is again like the CLARK method, but in **Minor** cycles, no deconvolution is performed, but only a selection of the strongest components down to some threshold. **Major** cycles are identical to those of the CLARK method. Although the principle is simple, the implementation is not easy because of normalization subtleties in the minor cycle stage. This method is reasonably well suited for more extended structures, but could become unstable if the threshold is inappropriate.

The Multi Resolution Clean (MRC, [Wakker & Schwartz 1988]) separates the problem in a smooth map and a difference map. Since the measurement equation is linear, both maps can be Cleaned (with Hogbom or Clark method) independently. This is faster than the standard CLEAN because the smooth map can be compressed by pixel averaging, and only fine structure left in difference map, so fewer Clean components are required.

15.7.4 The GILDAS implementation

All the above variants are implemented in the GILDAS software. All of them, except MX, are implemented both as tasks and as interactive commands in the MAPPING program. The later implementation allows definition of a flexible support constraint. The default method is CLARK. SDI & MRC are usually not necessary for Plateau de Bure, because of the small ratio between the field of view (primary beam) and the resolution (< 30).

MX is implemented only as a task, and not recommended because of its relatively slow speed. Since Plateau de Bure images are relatively small (128×128), it is easier to use a standard clean on larger images.

The GILDAS software does not include any implementation of the Maximum Entropy Method, MEM. The main reason is that MEM is not suited for limited uv coverage. But MEM also has some undesirable properties, among which its attempt to give a unique solution, with no physical justification, the noise dependent resolution, and the definition of a global criterium for adjustment to data. Furthermore, no noise estimate is possible on MEM deconvolved images.