Chapter 4

Algebraic Deconvolution

"This chapter is devoted to the use of linear algebra applied to the image restoration problem." (Andrews and Hunt, 1977, p. 147)

This quotation begins Chapter 8 of the rather prophetic book, *Digital Image Restoration.* The quotation served the original authors well; this was the book which largely introduced the modern concept of analyzing image restoration problems in terms of linear algebra. As the quote applies as well to my chapter as to theirs, I give their opening sentence one more use, some 18 years later. It is meant to underscore the fact that there is little fundamentally new in what I am about to describe. Indeed, singular value decomposition is a device from classical linear algebra, even if the first numerically stable algorithm to solve for the decomposition on a digital computer dates back only to 1965, (Golub and Kahan, 1965). The NNLS algorithm for the constrained least squares problem which we will make heavy use of later is from a book of a similar era, (Lawson and Hanson, 1974).

If the techniques are not new, then what is?

There is one fundamental distinction between the methods of Andrews and Hunt and those used here: In their methods, linear algebra was primarily used to analyse the problems — but most often the problems were reduced by clever manipulation of the matrix representations to more tractable ones in the frequency domain. Rarely were the problems solved directly, as the systems of equations were too large. While occasionally mention was made of the particular problems found in astronomy, it was clear that the authors were thinking in terms of the more usual picture processing. With every pixel filled with flux in varying degree, the general restoration such images is very difficult and one must resort to approximation, statistical treatments and cleverness. By contrast, here we will blunder through the problems using relatively simple and direct methods. Like the rest of this dissertation, the thrust of this chapter is largely pragmatic. There will be very little detailed analysis of the methods, some general discussion of why they work, and several examples demonstrating that NNLS in particular works *very* well for some problems. The previous chapter on weighting was a fairly thorough treatment. A few simple ideas were taken essentially all the way to their logical conclusions. Most of the obvious consequences were examined at least briefly. In comparison to that work, this chapter is a very promising beginning, but most emphatically far from complete.

Interestingly enough, there seems to be a convergence of research in the astronomical community towards algebraic methods of deconvolution, coming from quite different directions. I believe this chapter to be the first application of NNLS deconvolution to radio-interferometric imaging. In this field, the images were already too large for the computers of the day when NNLS was popularized as a model-fitting technique. It has only been recently realized that the overlap between computationally feasible images and scientifically interesting sources requiring high precision was large enough for algebraic techniques to be useful. The radio community has had to look backwards somewhat. The images that can be processed in this way are fairly small by the standards of the field, and the formalism is in the somewhat unfamiliar terms of model-fitting.

The high-energy astrophysics community is developing from the other direction. In the not too distant past, their detectors were of such low angular resolution that model-fitting was the only sensible type of processing to be done, and it was the image processing problems that were unfamiliar. To someone raised in a culture used to sophisticated modelling of complicated instrumental response and biases, it would seem only natural to continue to use such tools as the experimental resolution increased and the detectors became true imaging systems. And now, unlike 18 years ago, modern computers are largely up to the task of direct solution of the convolution equation. It will be interesting to see how similar the solutions become, starting from such different beginnings.

4.1 Representation

Representation issues were discussed in Section 2.3, where the continuous equations $I^D = I * B$ were forced into agreement with a dirty model representation $C * B$ at a collection of discrete points in the image plane, $\ell_{I\,i}$, for $i = 1 \dots N_\ell$. A component-based representation of C was arbitrarily selected, using scaled and shifted replications of an arbitrary pixel function P. The locations of the model components are $\ell_{C\,j}$, for $j = 1 \dots N_C$. (The usual form of the model chooses $P(\ell) \equiv \delta(\ell)$.) The convolution of C with B yielding I^D was discretized into a linear system of the form $\mathbf{Ax} = \mathbf{b}$. In matrix form, the elements of the matrix $\mathbf{A}$ are samples of the dirty beam B, the elements of the vector $\mathbf{b}$ are samples of the dirty image I^D, and the elements of the vector $\mathbf{x}$ are the magnitudes of the delta-function model components of C. If the generalized pixel representation is used, the $\mathbf{A}$ matrix is formed instead from samples of the modified dirty beam $B' = P * B$ and the unknown vector $\mathbf{x}$ contains the coefficients of the pixel function rather than the magnitudes of the delta functions.

In nearly all interferometric applications, the matrix $\mathbf{A}$ will be singular. There will either be none or infinitely many solutions to the system $\mathbf{Ax} = \mathbf{b}$, depending on whether or not $\mathbf{b}$ is in the range of $\mathbf{A}$. (In normal use, $\mathbf{b}$ will in the range of $\mathbf{A}$ by construction, though roundoff or approximation error in the calculation of the dirty image and beam may violate this at a very low level.) The problem of deconvolution, then, becomes how to select an appropriate solution from the infinitely many that satisfy or nearly satisfy the only equations that we have.

The only possible way to do this is to introduce additional information or beliefs that we have about what a "correct" answer looks like. As argued before, the only connection between I and C is that they solve the same convolution equation and yield the dirty map when convolved with the dirty beam. There is no particular guarantee that I will even be representable in the particular form we have chosen for C, and in general it will certainly not be so. We must use caution when applying physical constraints on the true sky to our model of it. But the constraints are all we have to guide the algorithm, so in general we will press forward and simply try to decide after the fact if some of our assumptions have been violated.

A particularly useful assumption is bounded support — the sky is mostly empty at radio wavelengths. This is probably the most important distinction between astronomical and conventional image processing. It is a strong regularizing operation on the problem, as demonstrated in a practical sense by the GSP algorithm. (Frequently, this algorithm will diverge when run with insufficient support constraints, yet when sufficient support constraints are given it can produce a high quality result. An example of this is given in Chapter 8.) As well as regularizing[1] the problem to some extent, bounded support also

[1] The term "regularization" is used somewhat loosely in this chapter. A formal definition of regularization

makes the problem considerably more tractable computationally when posed as a linear system. In the usual $\mathbf{Ax} = \mathbf{b}$ form, we have one equation for every location where we choose to enforce agreement between the dirty model and the dirty image, N_ℓ locations in total. At each of these locations, we must evaluate a sum of N_C model components. Bounded support, under the slightly questionable assumption that the model will be nonzero in locations where the image is nonzero, means that we do not have to include regions of the model where we believe a priori that there is no emission. In the matrix representation of the linear system, this corresponds to striking out columns. There is also nothing that compels us to use the equations generated by all the pixel locations in the image plane. If we believe that we have enough information, we may also arbitrary drop some of the equations, and choose to work with a subset. In general, more equations would be better, but we must make some concessions to the realities of computation. I call the collection of pixel locations which generate the equations we solve, the data window. This is all the raw data that the deconvolution algorithm will have to work with. The collection of places that we allow the algorithm to place flux is the flux window, and is much the same as any ordinary deconvolution support constraint. The beam matrix that connects the N_{flux} values in the component model vector with the N_{data} values in the dirty image vector is of size $N_{data} \times N_{flux}$.

Until now, I have been deliberately vague about the exact locations of model components and dirty image samples. The linear algebraic approach lends itself well to flexibility in discretization of the problem, and one can easily imagine applications that take advantage of this. One could place more model components in regions where the source varies quickly, for example. The algebraic formalism is also easily generalized to space-variant pixel models and point spread functions, though this is not used in this dissertation. The formalism then begins to converge with the Pixon approach of Puetter and Piña (1994) which does indeed seem a productive line of research. For our purposes here, the full-sized dirty image, dirty beam, and model components are all sampled or spaced at regular intervals on a grid,[2] $\ell_{Ii} \equiv i\Delta\ell$ and $\ell_{Cj} \equiv j\Delta\ell$. $N_C \equiv N_\ell \equiv N$. The discussion here will use a one-dimensional image for clarity, though the formalism is easily extended to two dimensions as shown by Andrews & Hunt (1977) and others. Inspection of the discretization

for the deconvolution problem is found in Rushforth (1987). A general discussion of regularization at a more generally accessible mathematical level is given in Press et al. (pp. 795–806, 1986). Essentially it means to take a problem which is ill-posed or noisy or both and select a particular solution by the introduction of additional information. Usually the process is controlled by a regularization parameter which determines the relative importance of agreement with the data and agreement with the additional information. The information may be in the form of an extra term in the optimization objective function, in which case the regularization parameter is simply a scalar multiplying the extra term, or it may be defined implicitly via finite iteration of an algorithm, in which case the number of iterations functions as a regularization parameter.

[2]Again we choose to work in the indexing system where $i = j = 0$ is the center of the image. Index arithmetic is assumed to operate modulo N_ℓ or N_C as appropriate.

equation 2.19 repeated here

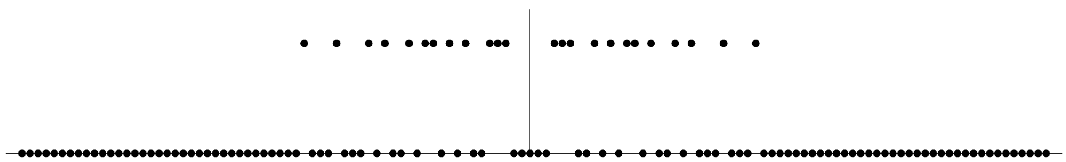
$$(C * B)(\ell_{I\,i}) = \sum_{j=1}^{N_C} C_j \,(P(\ell) * B(\ell))(\ell_{I\,i} - \ell_{C\,j}), \tag{2.19}$$

shows that for the usual case of $P(\ell) \equiv \delta(\ell)$ the elements of the full sized beam matrix are just $B_{i\,j} = B((i-j)\Delta\ell)$, which is Toeplitz in form and symmetric due to the symmetry of the beam. When using the discrete Fourier Transform of length N to calculate the images, the symmetry of the beam and periodicity of the transform produce the additional symmetry in the discrete beam $B(i\Delta\ell) = B((N-i+1)\Delta\ell)$.

The periodicity of the transform is sufficient to ensure that the full beam matrix is a circulant, where a circulant is a matrix determined entirely by the first row. Lower rows are simply the right shift by one element of the row directly above. Circulants are desirable matrices in that they can easily be diagonalized by the Fourier Transform matrix. The singular value decomposition of a circulant is similarly trivial to write down. Consequently, many techniques of algebraic image restoration involve ways of manipulating the problem until the beam matrix can be transformed into or approximated by a circulant or block circulant matrix.

That said, here we will do no such manipulations. We wish to encode the a priori support information about the problem into the beam matrix itself. The way to do that is exactly what we described above in the context of the linear equations. We pretend the pixels we do not want in the model do not exist, and strike the columns out of the beam matrix and the corresponding elements out of the model vector. We retain as many equations as we can, preferably in region where there is significant flux, and drop the rest. That is, we strike out rows of the full beam matrix and elements of the dirty image vector. We are right back to where we were before we assumed a regular grid, with an arbitrary sized matrix of no particular symmetry. We have given up the freedom to place flux anywhere we like, to no particular gain in computation. Clearly the flexibility of the approach is not being fully exploited here! But it is still there waiting to be used in future work.

An example of a one-dimensional beam matrix is given in Figure 4.1. (The other panels in the plot will be discussed momentarily.) The one-dimensional sampling which generated this beam is simply



and the corresponding one-dimensional PSF is

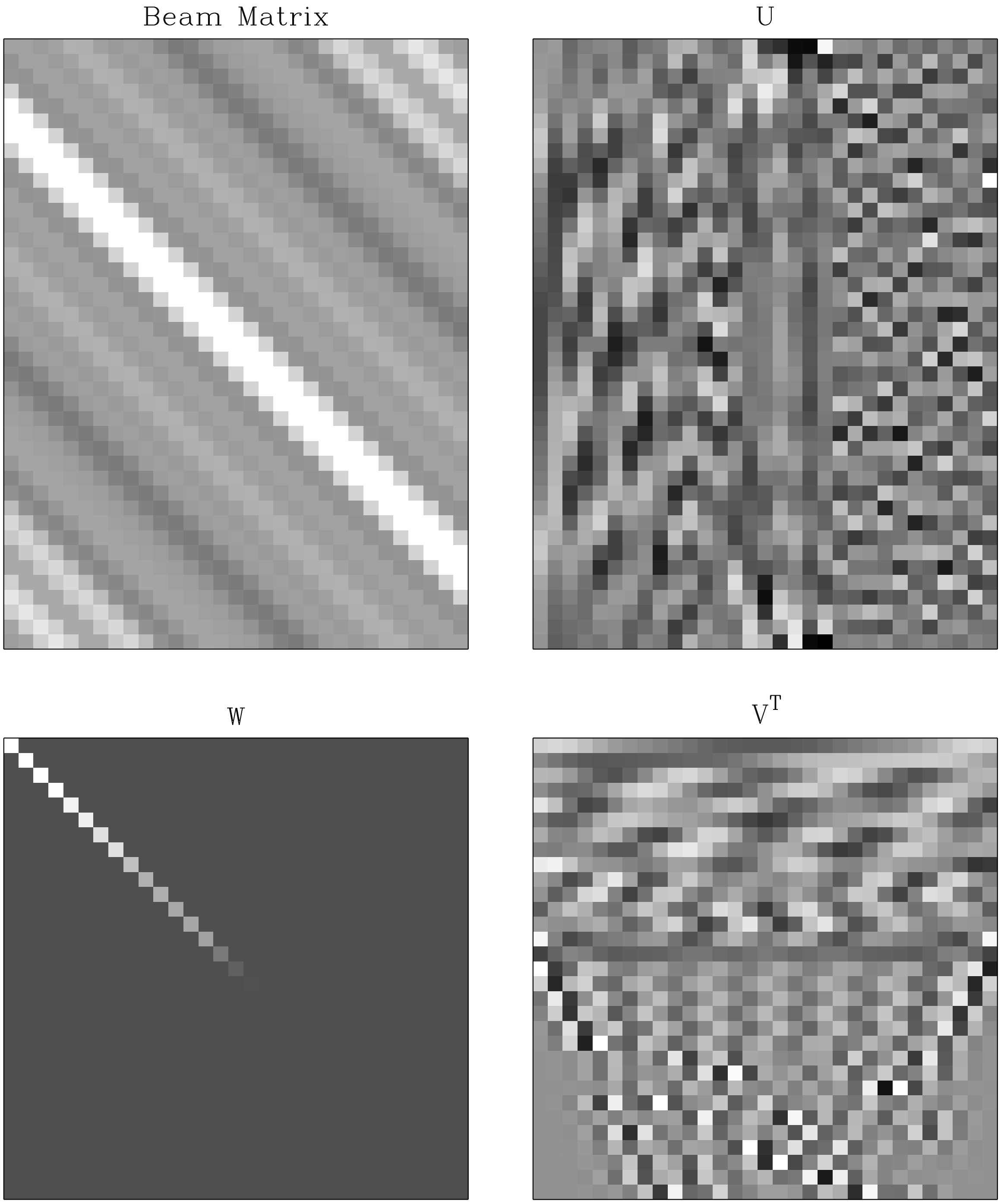
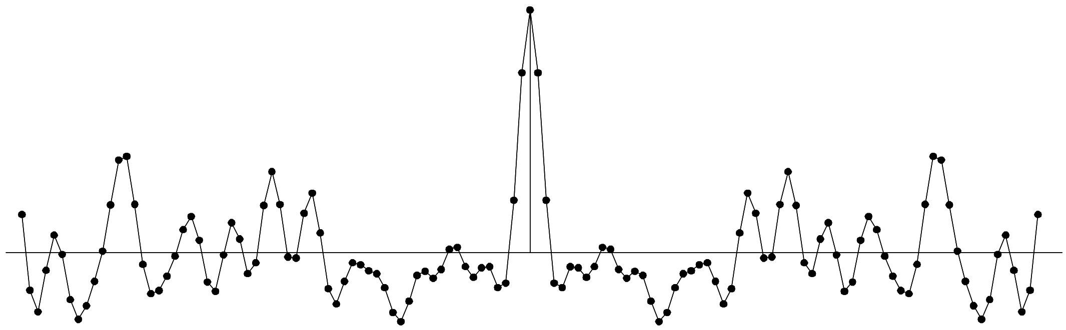


Figure 4.1: Singular Value Decomposition of a one-dimensional PSF



The data window was a 41-pixel-long segment centered at the origin, and the flux window was a 31-pixel-long segment also centered at the origin. In order to work with two-dimensional images in one-dimensional vectors, we merely need a stacking operator. In SDE, there an subroutine which reads an arbitrary portion of an image into a vector. It simply performs a raster scan of the image, left-to-right, and then top-to-bottom, copying the appropriate pixel to the vector whenever the corresponding pixel in the mask image is true. There is the inverse routine which maps a vector back into a masked image. And there is a routine which takes two masks and a beam image and produces the beam matrix connecting them. This is the only geometric knowledge that the deconvolution program needs. Everything else is encoded in the beam matrix. Figure 4.2 shows what a two-dimensional beam matrix looks like. It is taken from the supernova project in Chapter 8, and maps a 24 pixel diameter circle at the center of the dirty map to another circle of the same size on the component model image. The beam values which fill it are from the uniform weighting trace on Figure 3.19.

We now have a fully realized matrix equation equivalent to the original convolution equation. What do we do with it?

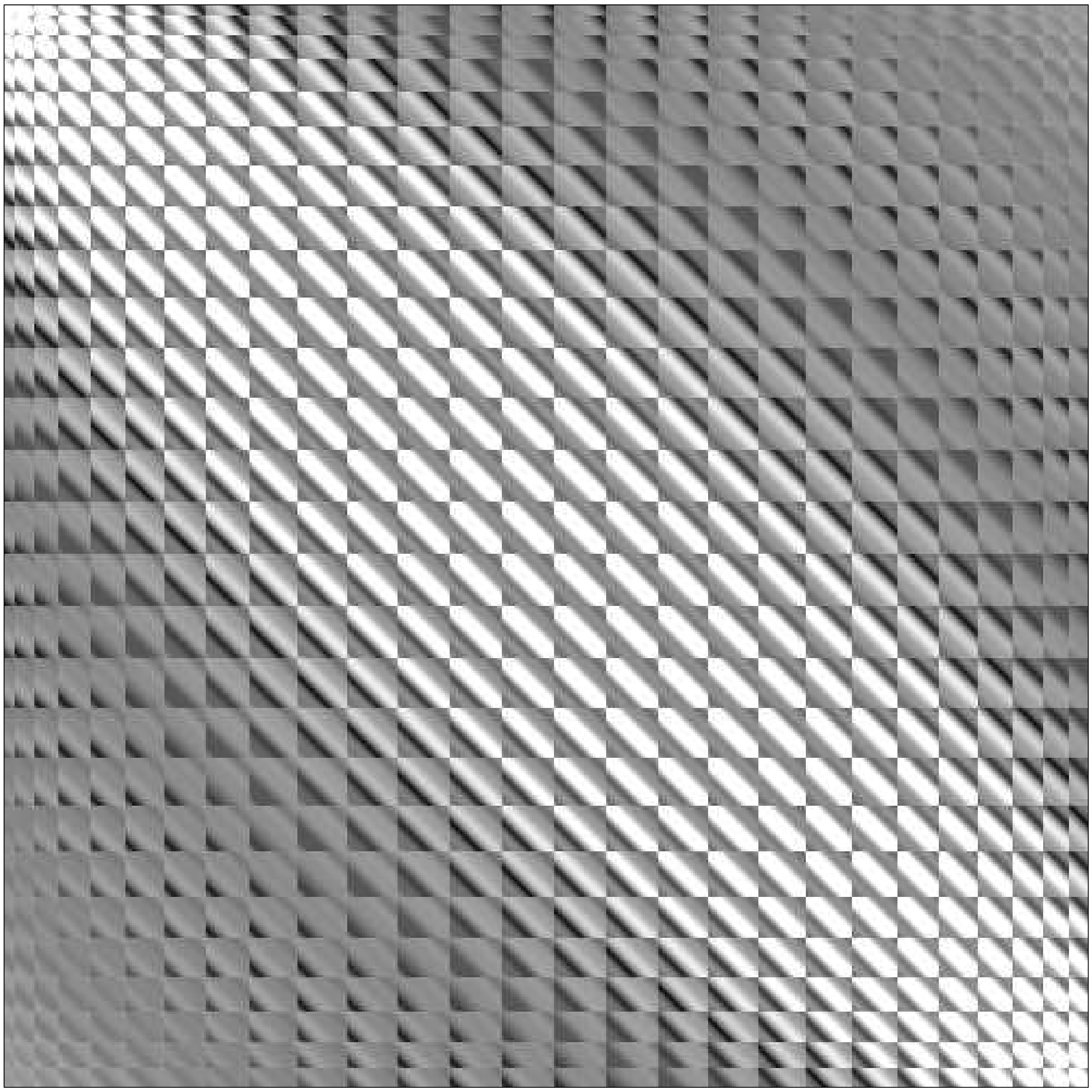


Figure 4.2: Two-Dimensional beam matrix.

4.2 Singular Value Decomposition

An obvious thing one might want to do with a matrix is invert it. Unfortunately, our matrix equation was derived from a singular integral equation, and will usually also be singular. A very general linear algebraic tool for dealing with singular or near-singular matrices is the Singular Value Decomposition, (SVD). It can be shown, (Noble and Daniel, 1977, p. 327), that any $m \times n$ matrix $\mathbf{A}$ with $m \geq n$ can be written as $\mathbf{A} = \mathbf{U}\,\mathbf{W}\,\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal and $\mathbf{W}$ is diagonal. (Minor modifications are needed to cover the case $m < n$, but we will not digress here.) $\mathbf{U}$ has the same $m \times n$ dimensionality as $\mathbf{A}$, (if $m \neq n$, then $\mathbf{U}$ is only column orthogonal), $\mathbf{W}$ and $\mathbf{V}$ are each $n \times n$. The orthogonality property means that $\mathbf{U}^T\,\mathbf{U} = \mathbf{V}\,\mathbf{V}^T = \mathbf{I}$. The elements along the diagonal of $\mathbf{W}$ are singular values of $\mathbf{A}$ and the eigenvalues of $\mathbf{A}^T\,\mathbf{A}$. They are hence guaranteed to be nonnegative. If the matrix $\mathbf{A}$ happens not to be singular, these properties mean that we can immediately write down the solution to the problem $\mathbf{A}\mathbf{x} = \mathbf{U}\,\mathbf{W}\,\mathbf{V}^T\mathbf{x} = \mathbf{b}$, namely

$$\mathbf{x} = \mathbf{V}\,[\mathrm{diag}(1/w_j)]\,\mathbf{U}^T\,\mathbf{b} \tag{4.1}$$

This expression makes manifest the dire consequences if one of the singular values happens to be zero and we naively try to solve the equation in this way. If one of the w_j does happen to be zero, it means that $\mathbf{A}$ is singular, and in that event there will either be no solutions to the equation or infinitely many. The SVD has the property, however, that if $\mathbf{b}$ is in the range of $\mathbf{A}$ so that there are infinitely many solutions, we may determine the particular value of $\mathbf{x}$ which has minimum length by replacing all the $1/w_j$ terms in equation 4.1 with 0 when $w_j = 0$, (Press et al., 1986, p. 54). If $\mathbf{b}$ is not in the range of $\mathbf{A}$, then the prescription above will yield the $\mathbf{x}$ that best minimizes the residuals $||\mathbf{A}\mathbf{x} - \mathbf{b}||$. Now it will usually happen that some values w_j will be very small. This corresponds to linear combinations of the singular vectors that are not well constrained by the data. In particular, they will be very sensitive to minor perturbations due to roundoff error or noise. Barring finite arithmetic, the residual norm is guaranteed to go down, the more singular values are included. But the reduction in residual norm caused by adding in negligibly small singular values will likely be tiny, and will be purchased at the cost of a wildly oscillating system of positive and negative values, all delicately cancelling out — if it can even be computed in finite arithmetic, of course! It is often better to discard more singular values than just the zeros. In general, one can use the number of singular values included in the solution for $\mathbf{b}$ as a regularization parameter, trading off agreement with the measured data for the stability of the solution.

Examine Figure 4.1 once again. Notice that the singular values along the diagonal of $\mathbf{W}$ have been arranged in nonincreasing order. The columns of $\mathbf{U}$ and rows of $\mathbf{V}^T$ that correspond to the larger singular values are the smoothest ones. These are the singular vectors of $\mathbf{A}$ and it happens pragmatically that when using the number of singular values as a regularizer, solutions with lower numbers of singular values are smoother. Now examine Figure 4.3. These are singular value spectra as the size of the flux window is varied. The size both of the data window and the dirty map is 128 pixels. A flux window of that size corresponds to an unwindowed decomposition. In this case, we can actually calculate the SVD decomposition analytically. One of the pleasant properties of circulants is that their singular values are the DFT of their top row, (Hunt, 1971). In the case of a dirty beam, the singular values are the gridded weights. This PSF was created with uniform gridded weights, so the singular values of the full PSF should be constant or zero. The entire long shelf of low values are spurious artifacts introduced in the decomposition. For numerical stability reasons alone, one must trim at a singular value threshold greater than 10^{-6} of the peak to remove these artifacts. Of course, after trimming away the spurious singular values for the unwindowed case, one simply re-obtains the original gridded weights. Multiplying by the singular vectors of a circulant is a DFT, so one gets back the uniform dirty map. (Even if the original weighting was not uniform, the gridded weights are divided by themselves, and one gets back the uniform dirty map.) If one throws away more singular values than the zeroes in the unwindowed case, one merely obtains another dirty map with fewer measurements. Clearly, in order to generate unmeasured spatial frequencies, one must encode some support information into the structure of the beam matrix. I do not know of a quantitative measure of how much information is included by tightening the support constraints, but it is worth noting that the fraction of significant singular values above the obvious cutoff grows as N_{flux} is decreased. I can say from a pragmatic standpoint that a window of 20 pixels out of 128 was sufficient to give reconstruction on a par or better than CLEAN.

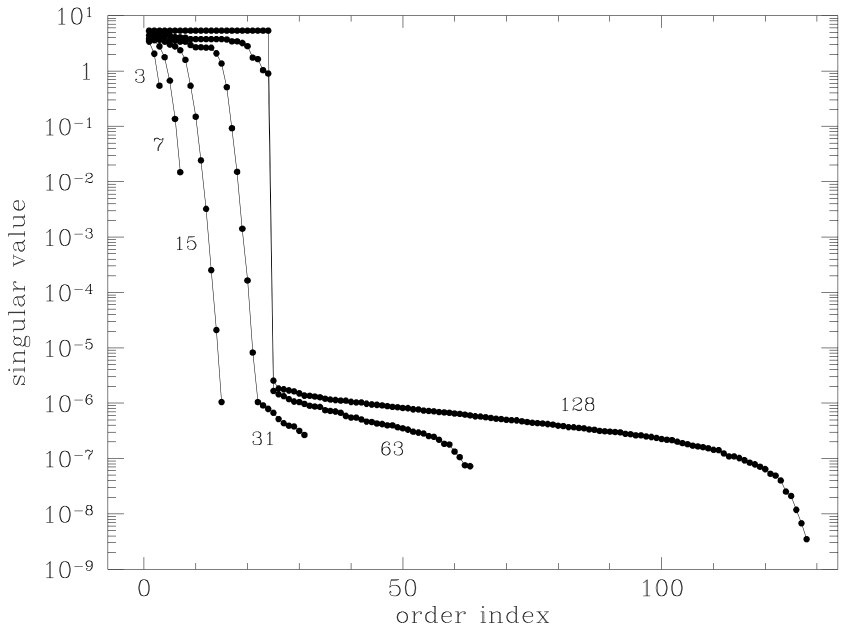


Figure 4.3: Singular value spectrum against window size.

The computational demands of direct algebraic methods are considerable. Several hours are required on an IBM RS/6000-580 workstation for an SVD deconvolution with

$N_{flux} = N_{data} = 3000$. The runtime scales with $N_{data}\, N_{flux}^2$, so there is a strong dependence on problem size. An even worse problem is memory, since that effectively sets a hard limit on the size of the problems which can be solved. At the least, one must hold a copy of the beam matrix in memory, of size $4N_{flux}\, N_{data}$ bytes, for normal real precision arithmetic. The algorithms examined for the SVD are not optimized for sequential memory access. If the beam matrix must be swapped to disk, the efficiency drops by a large factor, and problem which were just barely practical by runtime become completely unfeasible. SVD is even worse than other algebraic methods, in that it requires both the **U** and **V** matrices to be in memory at the same time. If the data and flux windows are the same size, this doubles the required memory over algorithms like NNLS which operate with a single copy of the beam matrix. Two different implementations of the SVD were examined, `svdcmp` from Numerical Recipes, (Press et al., 1986) and `SGESVD` from LAPACK, (Anderson et al., 1992). While the LAPACK routine was somewhat more robust than the Numerical Recipes routine, (the NR routine would occasionally fail to converge), the solutions produced by both were quite comparable. A more telling difference was between the single and double precision versions of these routines, and the double precision solutions were clearly superior.

I do not wish to belabor the SVD overlong, since by computational requirements and quality of solution both, it is inferior to NNLS. But it has been worth mentioning as a direct demonstration of how support information generates unmeasured spatial frequencies. And it can work reasonably well, if a human is examining the output and exercising judgement about where to make the singular value cutoff. Closing this section, I show two model deconvolutions of the usual uniformly-weighted 3C48 Gaussian model source. Comparing these reconstructions to Figure 6.11, we find that the single precision deconvolution is slightly worse than the CLEAN reconstruction, and the double precision deconvolution slightly better. Certainly the improvement in solution is not worth the dramatic increase in computation. The SVD by itself is interesting, but not worth pursuing unless it can be combined with another algorithm that needs the stabilizing properties of the SVD.

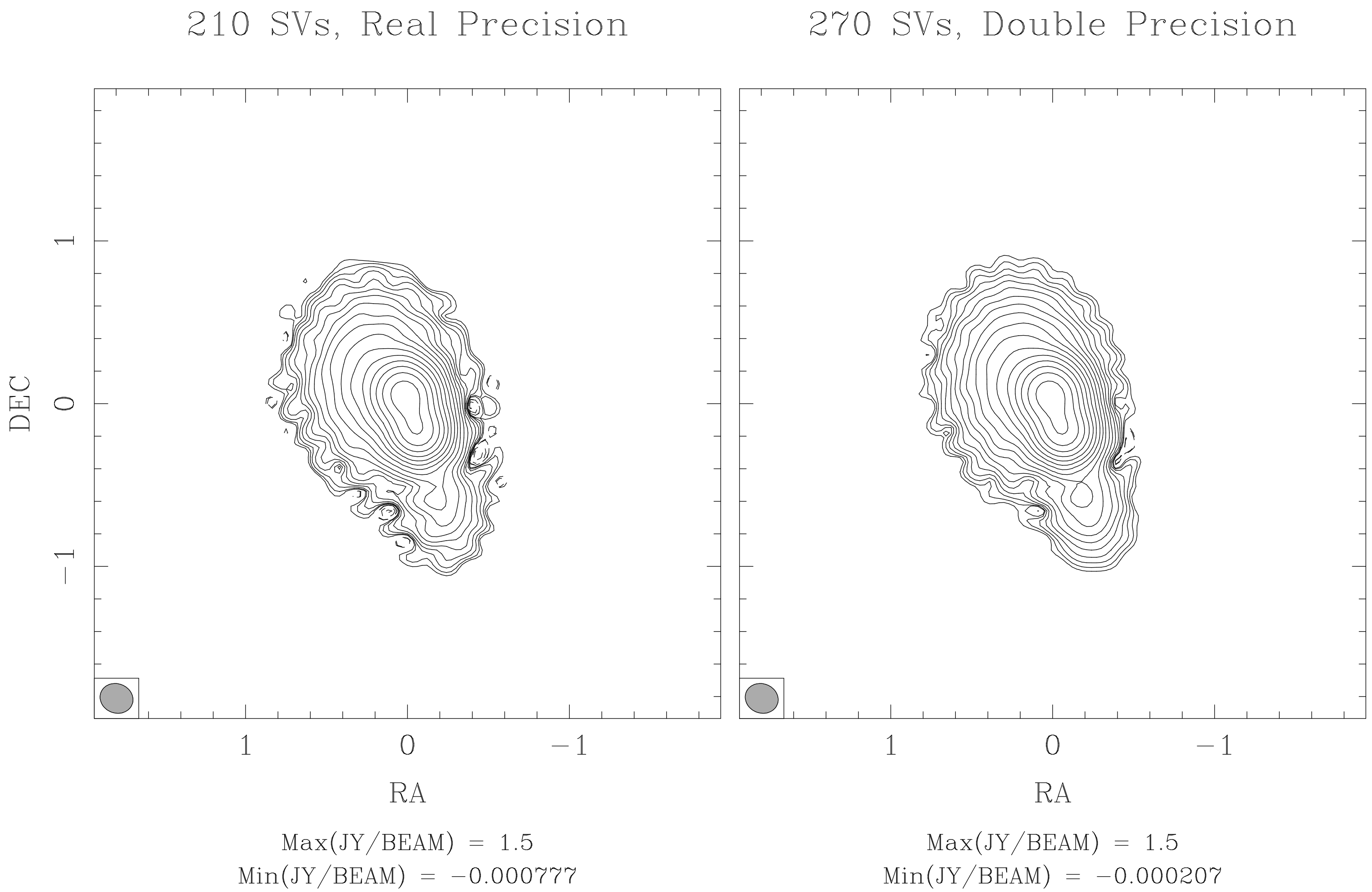


Figure 4.4: SVD deconvolutions of 3C48 model source. The left panel used the Numerical Recipes routine, `svdcmp` in real precision with 210 singular values included in the solution. The right panel used the double precision LAPACK routine `DGESVD` and included 270 singular values. Both use the manually selected optimum number of singular values for the reconstruction. Lowest contour is 100 μJy/beam, and there are 4 logarithmic contours per decade.

4.3 NNLS

SVD deconvolution happened to be the original motivation for the algebraic investigation. But once the basic framework was in place, it was a simple matter to try another algorithm which also purported to solve the basic matrix equation. The algorithm, NNLS from Lawson & Hanson (1974), also had the considerable advantage of enforcing positivity on the solution. This experiment was performed while working on a project to image the recent supernova SN1987A, and it was naturally applied there first. The project had a low signal-to-noise ratio, and was relatively simple to deconvolve at the nominal resolution. NNLS, like most other algorithms, produced an acceptable solution. The slipping point was that the project required the highest possible resolution and hence direct examination of the deconvolved model components. As will be shown by example later, the NNLS components do not appear terribly physical when examined directly. They tend to be spatially isolated from each other and spikey to the eye. CLEAN shares this behavior to an extent, and neither CLEAN nor NNLS was deemed suitable for the needs of the SN1987A project. Due to other concerns, the implementation was largely forgotten until the time came to write up this chapter as "an interesting exercise but ultimately not a particularly useful one." At this point, I resurrected the program and finally tried it on the model 3C48 source which had given so many other algorithms difficulties; I was shocked at how well it did! After subjecting the algorithm to the same suite of algorithmic test cases as the others, or at least as many of them as could be accommodated within the computational limit, I was eventually convinced that it really was performing as well as the initial tests had indicated — a significant advance in high dynamic-range imaging. The simulations did indicate that the NNLS solution degrades with source size compared to other algorithms, MEM in particular, and that the crossover point where it became comparable to other algorithms was somewhere near the limit of what could be processed computationally. The size scales at which NNLS excels are sources which are too compact for MEM to do a good job, and which are too extended for CLEAN to process well. Since the latter limit is surprisingly small, there are a significant number of scientifically interesting sources for which NNLS or a derivative algorithm is the tool of choice.

In an especially apt case of timing, the NRAO Very Long Baseline Array was just being brought up to full capacity. For the first time, a VLBI array was producing observations theoretically capable of being imaged at many tens of thousands to one dynamic range. And the images being produced were suffering exactly the same CLEAN artifacts as demonstrated in Chapters 5 and 6! NNLS deconvolution was attempted on the test observation of the bright compact source DA193, and it worked quite well as shown in Chapter 9. In the process, another useful property of NNLS deconvolution was discovered — it works particularly well as part of the hybrid mapping loop, where the data are iteratively imaged and self-calibrated. It was found that the interaction of CLEAN and self-calibration was limiting the composite process to somewhat lower dynamic range than CLEAN was capable of on perfectly calibrated data. In contrast, NNLS has consistently produced images of simple sources down to the thermal noise limit.

4.3.1 The NNLS algorithm

The reference for all of this section is Chapter 23 of Lawson & Hanson (1974). It provides not only the theory, but also the `FORTRAN` code that is the heart of the deconvolution program. I will describe the working of the algorithm in general terms, but it is both redundant and beyond the scope of the dissertation to do more than that.

The general class of problems to which NNLS belongs is the least squares problem with linear inequality constraints. That is, let $\mathbf{E}$ be a $m_2 \times n$ matrix representing the experimental design or deconvolution problem, $\mathbf{f}$ be the m_2 element data vector, $\mathbf{G}$ be an $m \times n$ constraint matrix and $\mathbf{h}$ be an m element constraint vector. The basic problem to be solved, Problem LSI, is

$$\text{Minimize } \|\mathbf{Ex} - \mathbf{f}\| \text{ subject to } \mathbf{Gx} \geq \mathbf{h}.$$

For our purposes here, $\mathbf{G}$ is the identity matrix, and $\mathbf{h}$ is identically zero. There is a condition which all solutions of the problem LSI must satisfy, called the Kuhn-Tucker conditions. Conversely, any vector which satisfies them is a solution. First note that the constraint equations can be interpreted as a series of m simple linear-inequality constraints. Every row of the constraint equation, $\mathbf{g}_i^T\mathbf{x} \geq h_i$ divides the allowed solution space in half with a bounding hyperplane. A given solution vector $\mathbf{x}$ can be either interior to, or on the bounding hyperplane for each row. The Kuhn-Tucker conditions say that the negative gradient vector of $\phi(x) \equiv \frac{1}{2}\|\mathbf{Ex}-\mathbf{f}\|^2$ points into the cone spanned by the negative normals to the constraint hypersurfaces containing $\mathbf{x}$. This means that in some appropriate coordinate system, the solution must have zero gradient in the directions where the solution is not hard up against a constraint surface. In the directions where it is on the surfaces, the negative gradient must point into the surfaces, not away from them.

It may help to visualize the case in Cartesian 3-space, where we wish to minimize $\|(x, y, z) - (0, -1, 1)\|$, while satisfying $y > x + 1$, $y > -x + 1$, and $z < 10$. The first two intersecting planes form an edge at $(x, y) = (0, 1)$, and clearly the point on that edge nearest $(0, -1, 1)$ will be the solution, namely the point $(0, 1, 1)$. The constraint that $z < 10$ is not coming into play, so there had better be no gradient projection along the z axis, or the solution cannot be a global minimum. The negative gradient is in fact $(0, -2, 0)$, which is normal to neither constraint plane, but "against" the sense of both.

The actual NNLS algorithm holds two index sets of indices, (or rather, one set and its complement), to indicate whether a given tentative solution element is zero (on a constraint surface) or not. Elements whose index is in the set $\mathcal{Z}$ are held at the value zero. Elements whose index is in the set $\mathcal{P}$ are free to take arbitrary values.

1. Set all elements of the solution vector $\mathbf{x}$ to zero, set all of the indices into the set $\mathcal{Z}$, set $\mathcal{P}$ to the empty set.

2. The main loop begins here. Compute the gradient vector $\mathbf{w}$ from the current value of $\mathbf{x}$: $\mathbf{w} = \mathbf{E}^T(\mathbf{f} - \mathbf{Ex})$

3. If $\mathcal{Z}$ is empty or if all elements of $\mathbf{w}$ with indices in $\mathcal{Z}$ have values ≤ 0, we have a solution. Terminate the algorithm.

4. Find the maximum element of $\mathbf{w}$, which will be positive. Move its index from the set $\mathcal{Z}$ to the set $\mathcal{P}$.

5. A possible secondary loop begins here. Form a modified version of the $\mathbf{E}$ matrix, $\mathbf{E}_{pos}$, where the columns corresponding to indices in $\mathcal{Z}$ are replaced with columns of zeros. Solve the unconstrained least squares problem $\mathbf{E}_{pos}\mathbf{z} = \mathbf{f}$. This will only determine the components of $\mathbf{z}$ corresponding to indices in $\mathcal{P}$. Set the remaining elements of $\mathbf{z}$ to 0.

6. If all the elements of $\mathbf{z}$ with indices in $\mathcal{P}$ are in fact > 0, this is an acceptable new trial solution. Set $\mathbf{x} = \mathbf{z}$ and continue with the main loop at (2).

7. We can only accept a fraction of $\mathbf{z}$ as the new trial solution: Find an index q such that $x_q/(x_q - z_q)$ is the minimum of all such expressions for negative elements in $\mathbf{z}$. For this q, call the expression α.

8. Form the linear sum $\mathbf{x} = \mathbf{x} + \alpha(\mathbf{z} - \mathbf{x})$

9. Move from the set $\mathcal{P}$ to the set $\mathcal{Z}$ all indices for which the corresponding element of $\mathbf{x}$ is zero. This will include q, and may include other elements as well. Continue the secondary loop at (5)

The computation of the subset unconstrained problem in step (5) is not actually calculated by filling unused columns with zeros, of course, but rather with an embedded QR decomposition that knows not to access elements beyond that allowed by the index sets. Also adding greatly to the efficiency of the implementation is that at each pass through step (5), we are solving nearly the same problem as before. The QR decomposition of $\mathbf{E}_{pos}$ is retained across iterations, and updated whenever an index changes sets. (Methods exist to update a QR decomposition in much less time than computing the entire decomposition from scratch.)

It is proved in Lawson & Hanson that the iteration of the NNLS algorithm is finite. Given sufficient time, the algorithm will reach a point where the Kuhn-Tucker conditions are satisfied, and it will terminate. There is no arbitrary cutoff in iteration required; in that sense it is a direct algorithm. It is not direct in the sense that the upper limit on the possible number of iterations that the algorithm might need to reach the point of optimum solution is impossibly large. There is no good way of telling exactly how many iterations it will require in a practical sense. The solution does improve smoothly as the iteration continues. If it is terminated early, one will obtain a sub-optimal but likely still fairly good image. Defining an iteration as the number of times the algorithm passes though step (5), Lawson & Hanson comment that their problems typically terminated after $n/2 = N_{flux}/2$

iterations. They hard coded a limit of $3n$ iterations into their code. I found it occasionally necessary to exceed that limit in model testing, and have allowed the maximum iteration number to be a free parameter. Certainly the algorithm is normally allowed to terminate on its own. The number of iterations required is also a function of the signal-to-noise ratio in the data. Noisy data means that the algorithm will try to fit negative flux to the noise power and will more quickly reach a situation where it cannot improve the solution by adding more components. The longest convergence times seen were roughly $10N_{flux}$, on large model objects that had been noiselessly sampled in the sky plane prior to creation of the dirty map. That is the, the most perfect possible model data set required several times Lawson & Hanson's hard coded maximum. All observational data required many fewer iterations, and none exceeded the original limit of $N_{flux}/2$.

4.3.2 How Important are the Windows?

This has not been investigated systematically on model data, but a reasonable amount of experience has been gained with the DA193 and 3C48 projects. It appears that the most critical factor is the size of the source itself, not the size of the window. On the DA193 project, both data and flux windows were varied by a wide range without significantly affecting the result. For the very highest precision deconvolutions, however, the window was edited by hand to exclude the most obvious spurious features at the window's edge. This made a small but noticeable difference in the solution. It would seem that a reasonable strategy is to deconvolve with as large a flux window as one can conveniently compute, using a data window at least as large as the flux window and larger if possible. Deconvolve once or as many times as needed if self calibrating, until one has a good calibration and reasonable image. At this point, direct examination of the component model image will likely show the main source and possible a slight scattering of very low level components off the main source. The latter can be excluded by hand for the highest quality image. It cannot be said that this procedure is excluding the existence of low emission at a level below that found by the wider windows, but one can claim that the image is consistent with no emission at the lowest level found by any means at all. (A fundamental principle of VLBI imaging has always been "If you see a scientifically interesting feature in your image, try hard to make it go away. If it does go away, it wasn't mandated by the data and you couldn't have trusted it anyway.") In comparisons on DA913, NNLS generally exhibited less sensitivity to support constraints than CLEAN, but was not completely immune to the problem.

4.3.3 Model Deconvolutions

Once again, we return to the 3C48 model source. Since this is shown several places elsewhere with uniform weighting, it was decided to use an intermediate robustness here. This was primarily to provide an example of the robust beams. Granted, this decision does favor NNLS somewhat, in that NNLS works particularly well with a robust beam, while CLEAN would prefer a uniform beam, but perhaps compensating in the other direction, we have used a model simulated in the visibility plane, and one that has significant image plane representation problems. CLEAN has reached the maximum fidelity it can on this source — issues of representation are not at all important for CLEAN here. A similar deconvolution where the model was discretized in the sky plane prior to creation of the dirty image yielded a nearly identical CLEAN image, apart from effects very localized to the region of the worst representation problems. By contrast, NNLS can go an additional factor of 10 deeper in off-source RMS when provided with such an ideal situation. So in selecting the parameters for this simulation, we have made one arbitrary decision that will favor NNLS and one that will favor CLEAN. As will be seen, the results are dramatic enough that it really makes very little difference how the details of the simulation are set up.

First, we show the raw and smoothed model in Figure 4.5. This is the same 3C48 model as used in the SVD section and the weighting chapter. Model parameters are given in Table 3.8, except that the magnitudes of the four outlier point sources have been adjusted to 1.19 μJy, 2.56 μJy, 5.51 μJy, and 11.9 μJy. The ratio between adjacent point sources corresponds to one contour in the following plots, the 3rd root of 10 ≈ 2.15.

Two support windows were used. The 'Large Window' is exactly that used in the CLEAN simulations presented in Section 3.10. It contains a fairly generous region for the central flux of 3C48, and circles of 11-pixel radius centered on each offset point source. This box is shown as a greyscale superimposed on the model plots in Figure 4.5. The 'Tight Window' is the same, but without provision for the point sources. The idea is that the astronomer would initially box the region of the visible source, and have no idea that the extremely faint sources are there. If the faint sources show up in the residuals, then they can be properly boxed and the deconvolution run again. The current computational limits to NNLS are enough for many practical problems, (roughly 6000 pixels of emission can be processed in a few hours on our IBM RS/6000-580 workstations), but they are not sufficient to allow boxing large regions on the sky just to see what flux might be there. If a faint source is to be detected, it must be initially detectable in the residuals. The pixel and dirty image sizes are the same as before, at $.03''$ and 512^2, respectively. The CLEAN used 100,000 components at a loop gain of 0.1, which was sufficient to reduce the residuals to 12.2 μJy/beam in all cases. Except where noted, all contours in this sequence are at the same fixed levels.

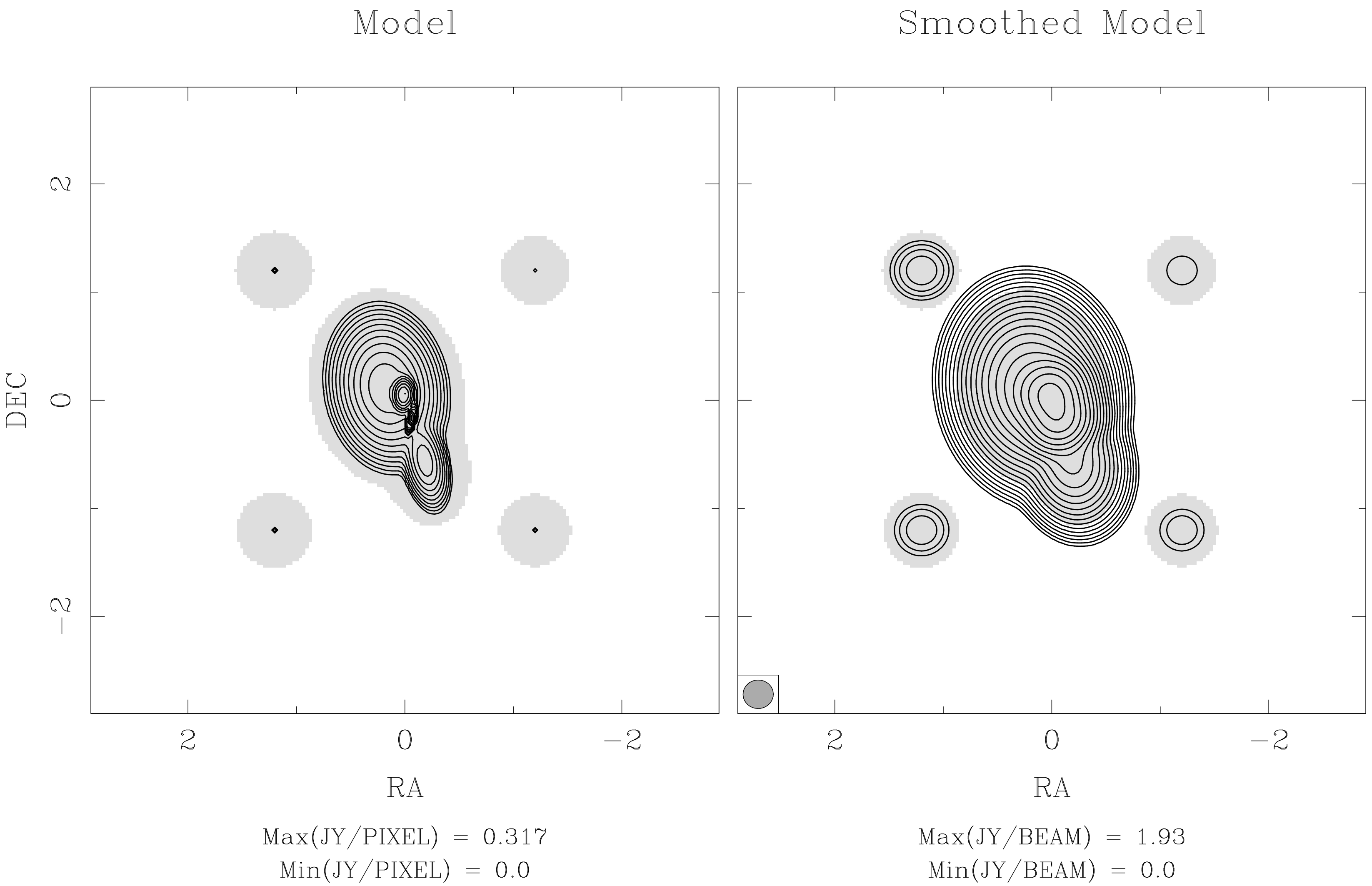


Figure 4.5: Raw and smoothed model source. The widows are the shaded region. The tight window is just the central shaded region, the full window is all grey regions collectively. Three logarithmic contours per decade. Lowest contour is .59 μJy/beam.

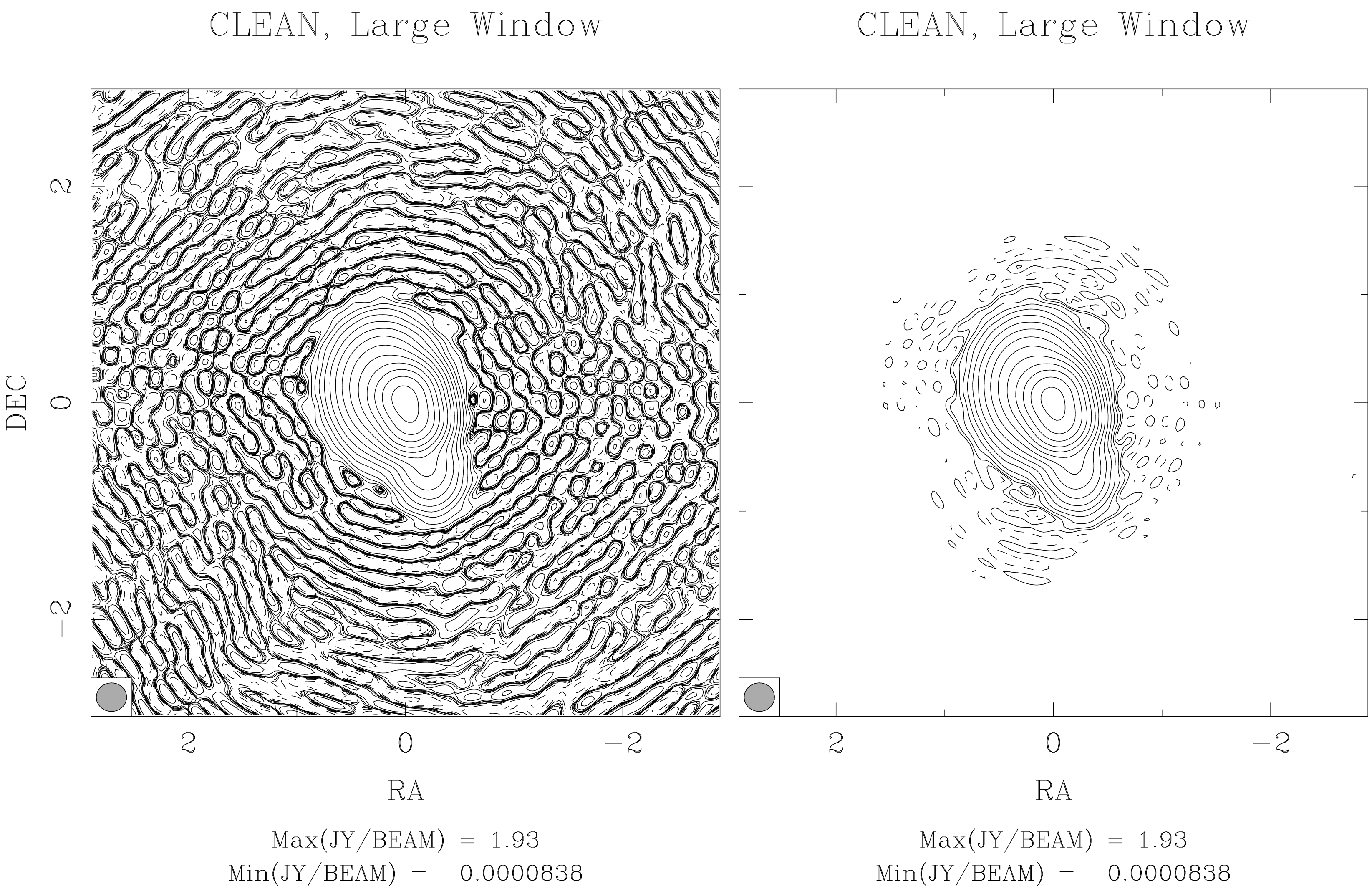


Figure 4.6: CLEAN deconvolution. Noiseless data, modelled in the visibility plane. The right panel is the same image, contoured up by a factor of 50. Lowest contours are .59 μJy/beam and 29.9 μJy/beam.

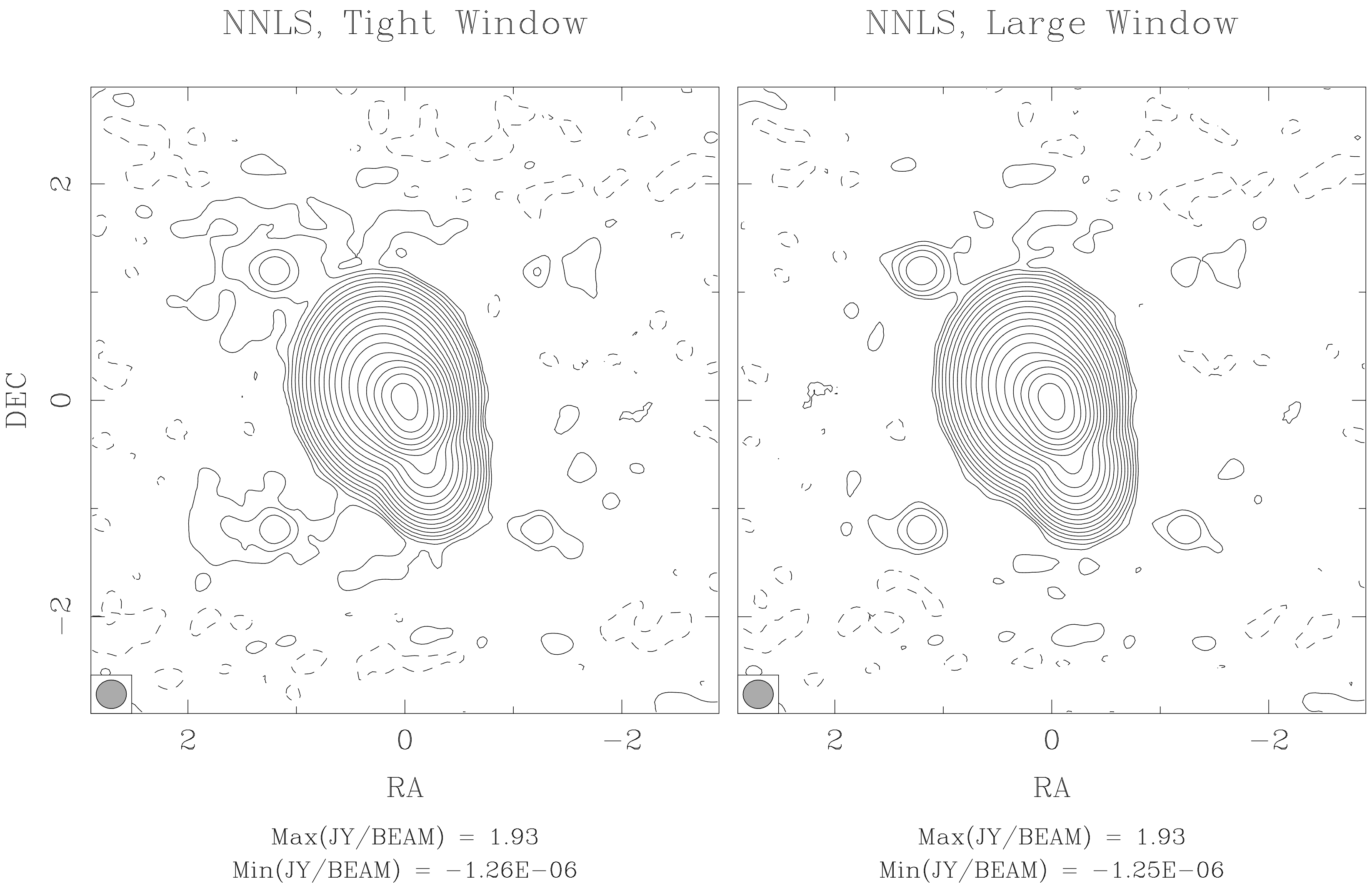


Figure 4.7: NNLS deconvolutions. The point sources are entering the image via the residuals in the left panel and via the components in the right panel. Lowest contours are .59 μJy/beam.

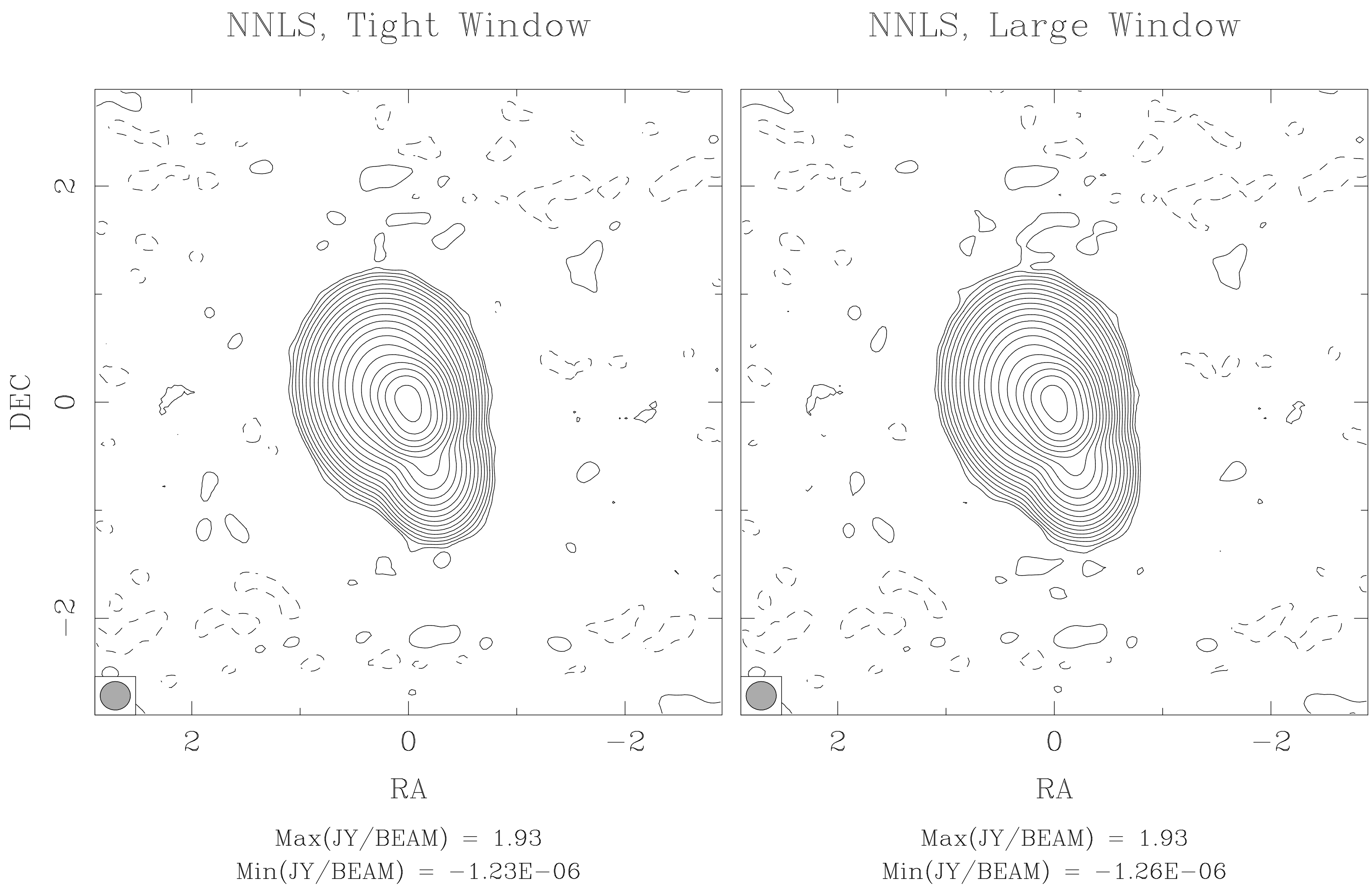


Figure 4.8: NNLS deconvolutions when the point sources have been removed from the model. Simply placing windows on empty sky is not sufficient to create a source.

Next we present the error characterization plots against weighting for NNLS. These parallel the similar plots for CLEAN and MEM in Figures 3.31 and 5.21, respectively. The simulation is identical to that described in Section 3.10, differing only by using NNLS instead of CLEAN. There are fewer points on the noiseless curve than in the other two figures, because the noiseless NNLS deconvolutions were quite computationally intensive, and only enough points to get the general shape of the curve were computed. Things to note about these plots are:

- The worse on-source error for NNLS is nearly independent of weighting in the noiseless case.
- NNLS has the strongest dependence of any of the major algorithms of full-image maximum error against weighting, with a fitted restoring beam. There must be a significant reconstruction error just outside the u-v sampling. CLEAN shows a similar qualitative behavior, if not as strongly.
- There is no significant difference between NNLS, CLEAN, and MEM on the noisy off-source maximum and RMS error cases. They all do pretty well.
- For the noiseless case, the difference between NNLS and the other algorithms becomes more apparent moving from maximum full-image to maximum off-source error, and still more so in terms of the off-source RMS error.
- The merging of the two lines on the noisy maximum off-source error plot indicates only that the worst off source error is a thermal residual that is not being smoothed, (and thus has no dependence on restoring beam). The curves converge on the noisy full-image maximum error plot when NNLS has dropped the on-source error low enough that the worst error is the same off source peak.
- In the noiseless case, the off-source RMS error has a minimum in robustness, right about where the worst near-in sidelobes of the beam have been minimized. While it is very well defined, it is not terribly significant in a quantitative sense, with the extremes differing by about 25%.
- It's interesting that the minimum in off-source RMS is not the uniform beam, which generally has the best far-off sidelobes of any weighting. CLEAN obviously prefers the far uniform beam in the noiseless case, and nearly has a maximum near the same robustness value.
- In the discrete model runs not shown here, the noiseless off-source RMS error for CLEAN was basically the same as in Figure 3.31. CLEAN had already bottomed out due to the algorithm and slight under-CLEANing; the details of the sampling made little difference. The same thing is nearly true for MEM. With the perfect sampling, NNLS reduced the off-source RMS error to of order $2.5(10)^{-8}$ Jy/beam, roughly 10^3 times deeper than CLEAN, and 10^4 times deeper than MEM!

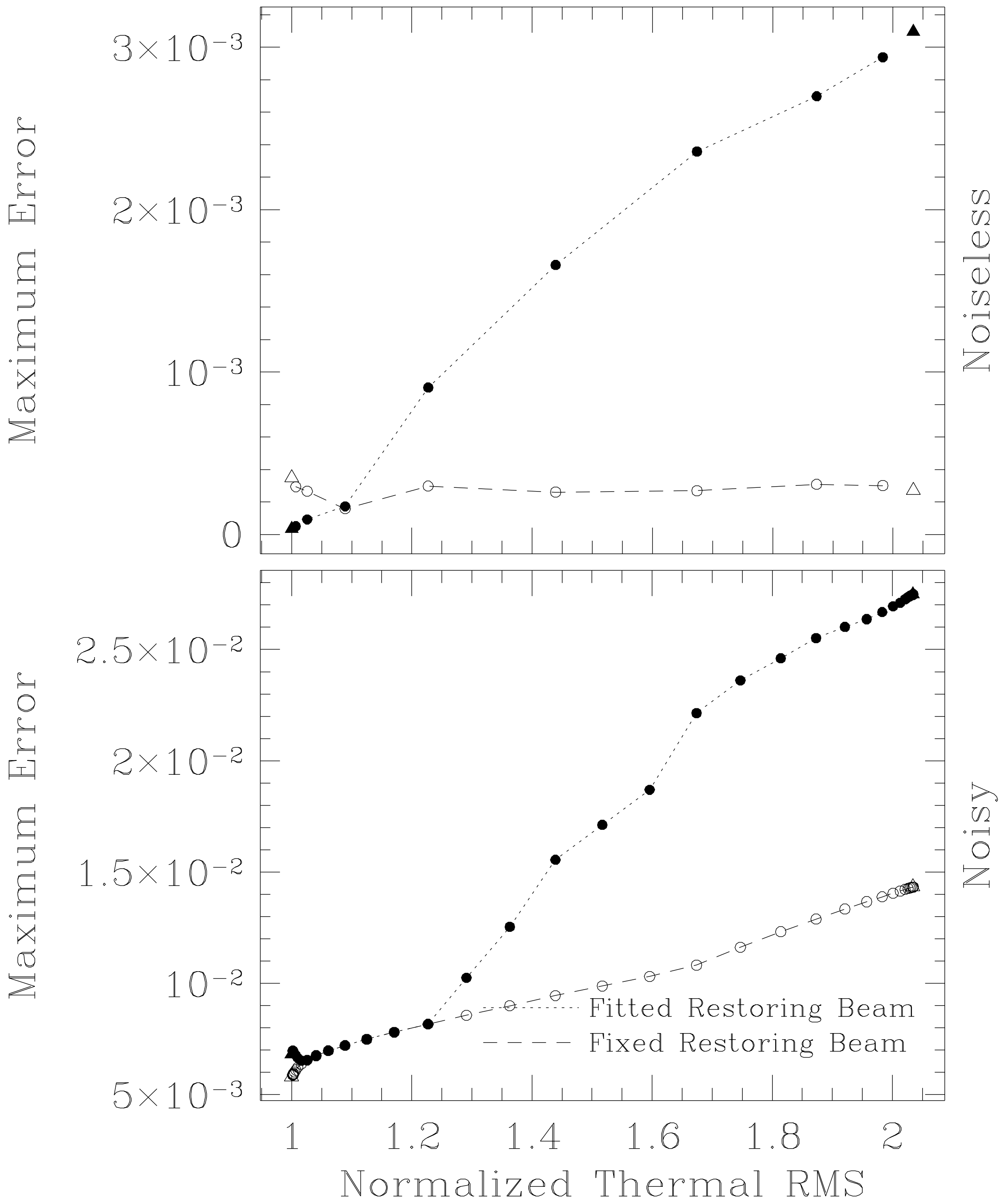


Figure 4.9a: Error quantities against RMS degradation factor: maximum error. The fitted restoring beam corresponds to Figure 3.30. The fixed beam curve is more representative of the absolute deconvolution quality and used $\sqrt{B_{nat}B_{uni}} = .2214''$.

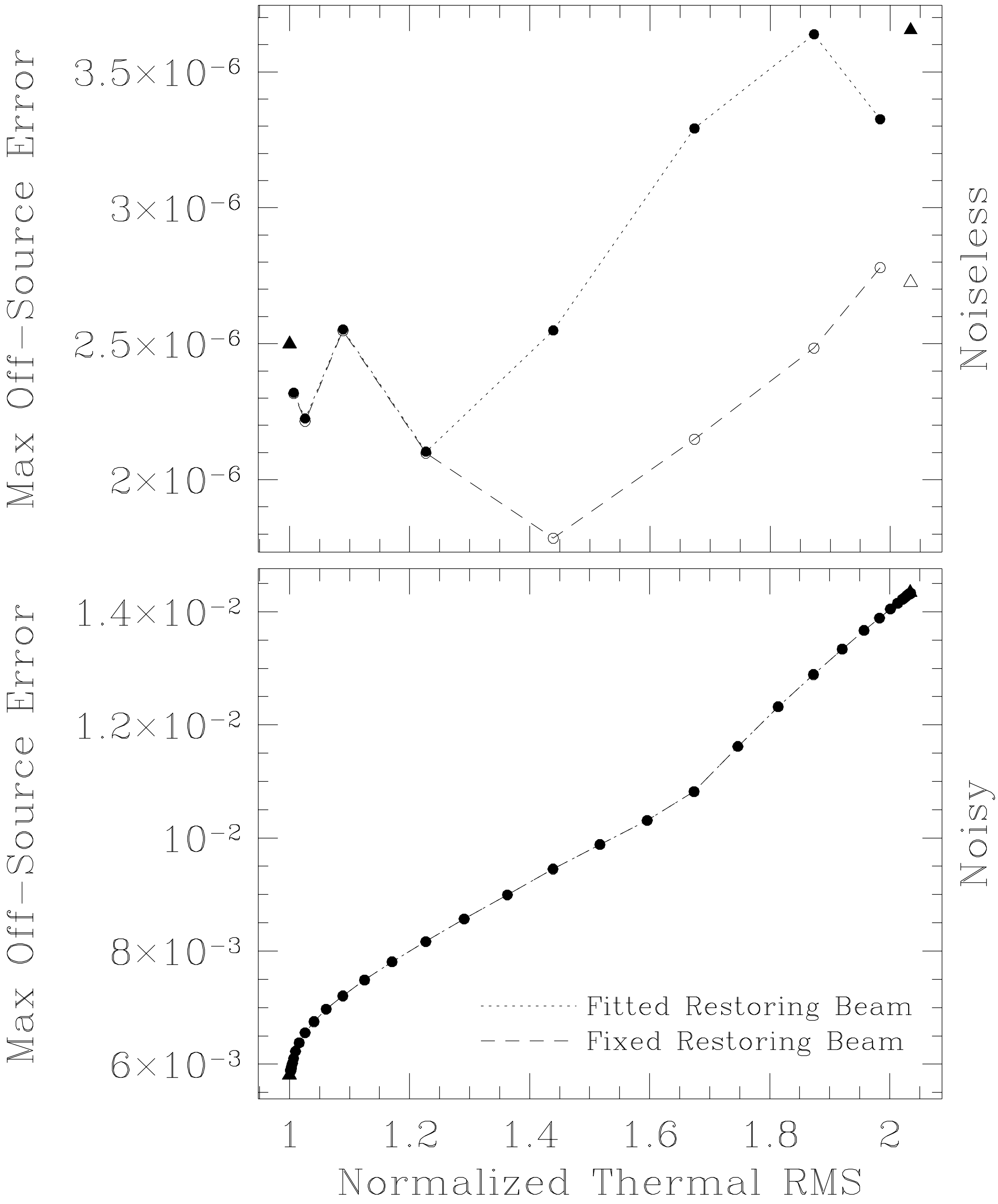


Figure 4.9b: Error quantities against RMS degradation factor: maximum off-source error. The two traces in the lower panel are identical.

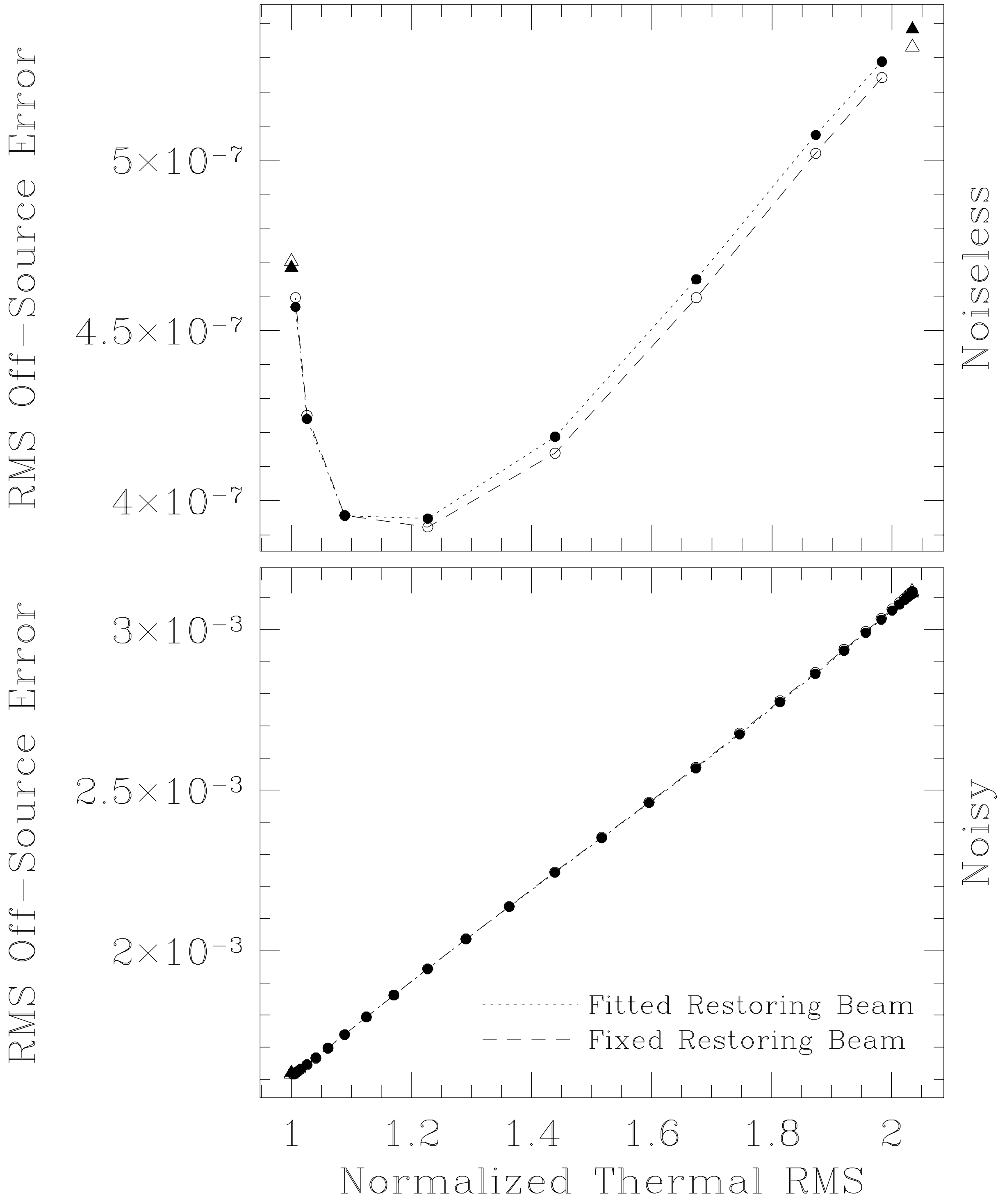


Figure 4.9c: Error quantities against RMS degradation factor: off-source RMS error.

Finally, we present one fairly amusing example both of the flexibility of the algebraic approach and of the robust nature of NNLS deconvolution. At an early stage in the image processing of DA193, the calibration was such that there was a slight extended halo around the source, which later turned out not to be real. For the moment we may take it as the truth image, since that is what is present in the data as far as the deconvolution algorithm is concerned. In fact, such a "truth image" is slightly unfair to NNLS, in that the image it is trying to recover has negative values which may partially disrupt the solution process.

Figure 4.10 shows two deconvolutions of this data set. In the first, the data window is the size of the large grey circle, with a radius of 38 pixels and an area of roughly 4500 square pixels. The flux window is the smaller grey circle, with radius 20 pixels and an area of 1320 square pixels. The left panel is the standard NNLS deconvolution using these windows, and for this demonstration may be considered what the source looks like. In the right panel, the flux window was unchanged; the algorithm could place flux in the proper location. The data window, however, was changed to be the annulus between the two circles. The windows are now completely disjoint. The algorithm was not given the dirty image at the locations where it was placing flux. It reconstructed the source entirely through the sidelobes. As can be seen by comparing features, the second deconvolution is not as good as the first, but it is remarkably close. NNLS is even quite capable of reasonable deconvolutions when there are more pixels in the flux window than in the data window. Obviously, this can only be sensible with the addition information introduced via the positivity constraint. I know of no way to quantify how much additional information is introduced, but in practice it seems that given a significant piece of the dirty image to work upon, NNLS will return a reasonable answer.

I have been calling this a "deconvolution party trick," since I can think of no good application of this ability in radio interferometry. Locating a far source by its sidelobes leaps to mind, but unfortunately the NNLS algorithm cannot image a large field without knowing where to place the support windows. In the optical or any image plane detection system, of course, one may have the problem of missing data due to detector saturation or to cosmic rays. This has not been pursued at all, but it seems like it could be useful.

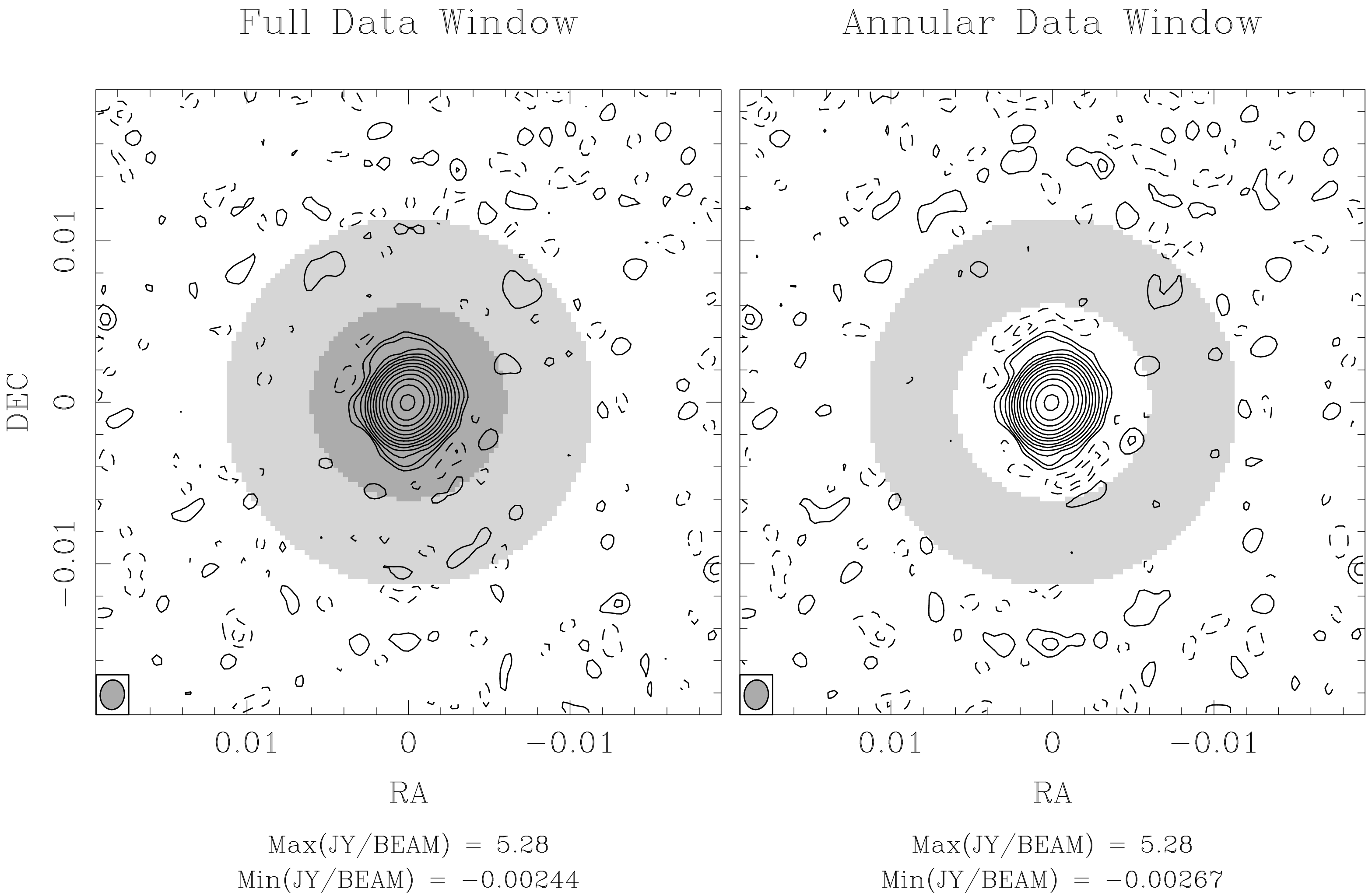


Figure 4.10: Missing data deconvoltion: On the left, the algorithm has been given all pixels on the dirty map interior to the large circle as input to the deconvolution. It was allowed to place flux within the small circle. On the right, it has been given only the pixels corresponding to the shaded annulus as data, but has reconstructed the source from the sidelobes alone.

4.3.4 Discussion

NNLS has plenty of limitations and some problems. But when it works, it works *extremely* well. In an absolute sense, it works best on extremely compact emission. Compared to the other algorithms commonly used in radio astronomy, it best distinguishes itself on sources too extended for CLEAN to process well, and too compact for MEM. It has no difficulty with sharp edges, which make it a good choice for planetary images and source involving a shock. It is also very good at producing models for self-calibration. Both of these properties are presumed related to the fact that NNLS zeros the residuals in the dirty image nearly completely. Some algorithms like MEM make no particular attempt to zero the residuals completely, and simply fit the data to within the thermal noise. Unfortunately, if the residuals are correlated in the image plane, this may result in significant systematic bias at different locations in the u-v plane — very bad news for self-calibration. CLEAN will indeed zero the residuals in the asymptotic limit, but at finite iterations may also suffer a systematic bias in the u-v plane. NNLS makes a choice about the model, and then essentially jumps right to the solution. This can be a good thing or a bad thing, depending on the quality of the data, but under-iteration is rarely a problem in NNLS images. And even with quite noisy data, the solution appears robust.

For practical purposes, the NNLS algorithm lacks a termination criterion, or any other tuning parameter. This is a good thing, since any parameter which must be set for proper operation can be set incorrectly. NNLS certainly exceeds the capabilities of CLEAN and MEM on objects to which it is well suited. But also, it has the advantage that it is difficult to misuse and generally produces the best image of which it is capable. The other algorithms can produce sub-optimal images due to insufficient iteration or poorly chosen parameters. A major strength of NNLS as a practical deconvolver is that it is largely "hands off."

On the negative side, NNLS shares the tendency of CLEAN to compact flux. A region of smooth flux will typically result in a model with regularly spaced components at intervals of a few pixels. The isolated pixels will contain all the flux that should be contained in the zero pixels surrounding them. This behavior seems to be a common failing of algorithms that operate on small numbers of pixels at a time.

The Kuhn-Tucker conditions do not guarantee a unique solution to the deconvolution problem, merely "a" solution. Clearly, since the performance of the NNLS algorithm on noiseless, discretely sampled model data is less than perfect, there must be multiple solutions present that even in the perfect case are indistinguishable by the algorithm from the real thing. Examinations of the residuals in such perfect cases show that it really has reduced them to within machine precision of zero. Unless this were so, the real solution would have a calculably better fit to the data and the algorithm would not terminate. Since the algorithm typically builds up the model a few components at a time, examining the termination criteria at each step, it will stop at the first solution that satisfies the Kuhn-Tucker conditions to its precision. In some sense, then, the NNLS algorithm is selecting something like a minimum-number-of-pixels solution. The addition of positivity and support has reduced the dimensionality of the null space somewhat, but the problem is still singular. The final implicit criterion in the algorithm is that it stops at the first acceptable solution, which is built up a few pixels at a time. It means that if the real solution has a great many smooth pixels, the algorithm will always select a flux compacted solution over the true answer, so long as both are indistinguishable from the point of view of fitting the data. It definitely would be worth looking into other algorithms that solve the same in different ways, to see if variations in how the solution is computed make a significant difference.

The NNLS algorithm is used in the simulation Chapter 5, the 3C48 case study Chapter 7, the SN1987A case study Chapter 8 and the DA193 case study Chapter 9, so we will see a good deal more of its properties by example. The SVD deconvolution is only used henceforth in Chapter 8.