A Comparative Study of some Deconvolution Algorithms

Urvashi Rau

June 25, 2001

1 Introduction

The problem of deconvolution, in synthesis imaging, can be solved numerically in various ways. It can be considered as a least squares fitting problem, with the parameters to be determined, being the pixel values of the actual brightness distribution. The most popular techniques are those of CLEAN and Maximum Entropy. CLEAN is an iterative algorithm that treats the image as a collection of point sources in the sky. MEM is an iterative least squares fitting algorithm. The objective here, is to obtain a solution as close to the original, as possible,i.e. a minimum norm solution.

Algebraic deconvolution methods like Singular Value Decomposition, and Non Negative Least Squares, by definition, produce minimum norm solutions.

Also, the convolution equation, can be considered as a set of simultaneous linear equations,(a matrix equation), with a very large number of parameters to be determined. In addition, the system may be either over-determined or under-determined, resulting in singular matrices and related instabilities. There is therefore the requirement for stable linear algebraic techniques to take care of the instabilities associated with ill-conditioned systems. Such systems become sensitive to constraints applied to the data and the algorithms. One prominent type of constraint, is the window constraint, which serves to regularize the problem,by restricting the algorithm to work on areas within certain specific areas in the image.

This short study is based on one iterative algorithm, and two algebraic deconvolution algorithms, the Singular Value Decomposition, and the Non Negative Least Squares. These algorithms have been implemented in a Matlab based Octave environment, on a one dimensional image. The results have been interpreted with the aim of emphasizing certain features of these algorithms, that may be exploited to develop optimized algorithms based on these basic techniques.

2 Deconvolution in synthesis imaging

The van-Cittert-Zernike theorem

$$V^0(u,v) = \int\int I^0(l,m)e^{-2\pi i(ul+vm)}dldm \tag{1}$$

describes a fundamental property of interferometry, which states that the brightness distribution in the sky, $I^0(l, m)$ is the two dimensional Fourier transform of the visibility function,$V^0(u, v)$, i.e. the two point correlation function on the ground. An interferometer, makes measurements at a limited number of positions on the visibility plane. This defines a sampling function in the visibility plane, also called the uv-coverage, and corresponds to measuring a limited number of Fourier components, or spatial frequencies of the source brightness distribution.

The measured visibilities can be given by $V^{obs} = V^0 S^v$ where V^0 is the true Fourier transform of the brightness distribution on the sky, and S^v is the sampling function in the visibility domain.

The direct inverse Fourier transform of S^v, gives a 'dirty image', and is given by

$$I^D = F(V^0 S^v) \tag{2}$$

By the convolution theorem, this corresponds to

$$I^D = F(V^0) * F(S^v) \tag{3}$$

$$I^D = I^0 * B \tag{4}$$

where B is the inverse Fourier transform of the uv sampling function, and is called the dirty beam. It is the transfer function (or point spread function) of the interferometer, and is the image formed when a point source is observed.

The dirty image, or the Principal solution, will possess only those spatial frequency components, as measured in the visibility domain. Values at the unmeasured spatial frequencies, have to be generated, to reconstruct the the original brightness distribution. This process is deconvolution, which aims to obtain as close an approximation to the actual distribution as possible.

The deconvolution equation $B * I^0 = I^D$ can be written as a system of simultaneous linear equations as follows

$$BM_{m\times n} I^0_{n\times 1} = I^D_{m\times 1} \tag{5}$$

where BM is the beam matrix, formed from the dirty beam. Each row of the BM is the dirty beam, shifted by one as the rows advance, resulting in a circulant and toeplitz matrix. The convolution operation is thus expressed as a matrix multiplication, and this can be appreciated from the formal definition of a convolution operation, as a sequence of shift-multiply-add, between two vectors.

Through this matrix formulation, the deconvolution problem is transformed into a set of simultaneous linear equations, which can be solved by algebraic methods. The number of rows of the beam matrix is the number of equations(number of points at which measurements have been made), and the number of columns, represents the number of unknowns(the number of pixel values that have to be determined in the reconstruction).

3 Simulations

Numerical solutions to the deconvolution equation ($I^D = I^0 * B$)have been attempted,by implementing the following three algorithms.

1. Iterative Deconvolution
2. Deconvolution by Singular Value Decomposition (SVD)
3. Deconvolution by Non Negative Least Squares (NNLS)

The simulations have been done for the one dimensional case, where the original 'source' function is the sum of three gaussians, evaluated at 256 regularly spaced points.

The sampling function in the visibility domain, is a symmetric set of ones and zeros, corresponding to a one dimensional cross section of the GMRT uv coverage. There is a significant gap in this sampling function, in the short-spacing visibility points. The original source function and the uv coverage used are as shown in Figure 1

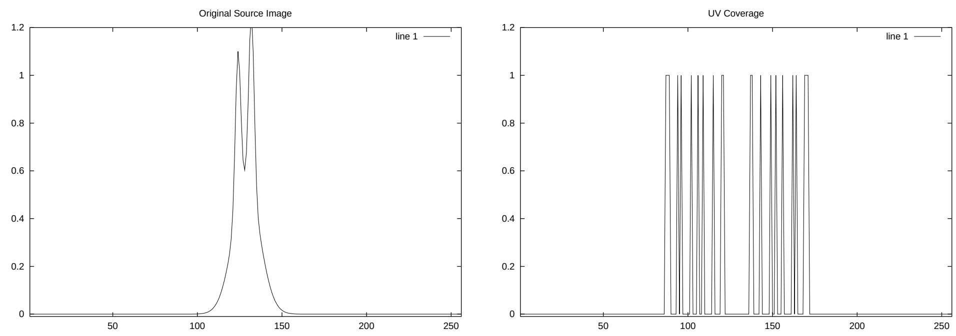


Figure 1: Original Image and UV Coverage

These algorithms work with a data window, and a flux window. The data window specifies the part of the dirty image, that is to be used for the deconvolution, and the flux window defines the interval within which the reconstruction should be confined. In the SVD and NNLS algorithms, the data window corresponds to the entire dirty image (all 256 values) and the flux window is an interval of specified length, symmetric about the centre of the 'source'. In the iterative algorithm, both, the data and flux windows, are limited intervals, symmetric about the centre of the 'source'.

For each algorithm, the original 'source' function, as well as the final reconstruction, are both convolved with a clean beam (a gaussian fitted to the central lobe of the dirty beam), so as to perform all interpretations at the right resolution. Deviations from the ideal solutions are calculated as the mean square difference between the original 'source' and the reconstructions, in both, the spatial as well as the visibility domains. These values are used to compare the accuracy of the reconstructions, as a function of window size. In the iterative case, the root mean square differences along with a suitable tolerance level, forms the convergence criterion.All simulations are essentially noise free, though the algorithms have been tested for minimal amounts of additive gaussian noise in the visibility domain.

The following sections describe each of the three algorithms in detail, along with implementation details and results.

3.1 Iterative Algorithm

This is a simple algorithm which alternates between the image and visibility domains, enforcing the flux window as a support constraint in the image domain, and forcing agreement between measured and generated values in the visibility domain.

An initial guess of the original 'source' function, is made to converge iteratively, to as accurate a reconstruction as possible.

3.1.1 Algorithm

The following steps are carried out for a particular flux window size, in the basic set of iterations.

1. The iterations begin by taking the Fourier transform of the initial guess, and replacing the visibility values at the measured points, by the measured values.

2. The inverse Fourier transform is taken to obtain a partial solution. In this image domain, the support constraint in the form of a window, is applied, by retaining only the values within the window and setting all values outside the window, to zero.

3. The Fourier transform is taken, which is now found to contain non zero values at the unmeasured points.

4. A convergence check is made, by comparing the accuracy of the current reconstruction, in both the image and visibility domains.

   In the image domain, the ratio of the total flux outside the window, to that inside it, is calculated.

   In the visibility domain,the root mean square difference between the generated values at the measured points, and the actual measured values at those points is calculated.

   These two values are compared with predefined tolerance levels, and unless both values fall below their respective tolerances, the iterations continue.

5. In order to impose a constraint of constant total flux, all the generated visibility values at the unmeasured points, are scaled according to the ratio between the mean square of the measured visibilities and that of the generated values at those same points.

6. Now the generated values at the known points are replaced by the measured visibilities. The iterations continue from (2), until the convergence criteria are met.

Different data and flux window sizes give reconstructions of different accuracy, and hence, a variable or adaptive window size would be desirable, to obtain the best possible reconstruction.

3.1.2 Implementation and Results

The initial guess is constant within the data and flux windows, and zero elsewhere.

The iterations begin with a window size, smaller than the extended region of the source, and are taken to convergence, at that window size. This gives a reconstruction only within the window. Now, the window size is increased, and the iterations are continued, with the result of the first set of iterations, as the initial guess, for the second. Now, because of the zero space errors in the uv coverage, it is possible that most of the extended structure in the source has been left out. In order to partially reconstruct this extended structure, a low constant is added to the result of the previous set of iterations, and this is used as the new initial guess. This additive constant can also serve as a sort of positivity constraint

These iterations at different window sizes are continued until the extended region of the source is sufficiently reconstructed. This implementation begins with a minimal window size of 30 pixels and stops after 3 iterations, at a window size of 70 pixels.

The basic iterations were carried out for window sizes varying from 20 pixels to 80 pixels, and a plot of mean square differences in the image as well as visibility domains, was obtained.(Figure 3.1.2)

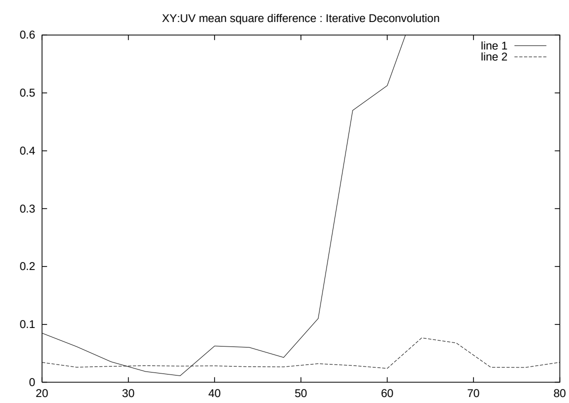


Figure 2: Plot of XY-UV-mean square differences for different window sizes

From the above plot, it can be seen that there is a small range of preferable window sizes, for which the deviations from the ideal, in both, the image and visibility domain are minimum. For large window sizes, the reconstruction tends to become unpredictable.

For this implementation, the following plots (Figure 3) represent the initial guesses, and final reconstructions at each stage of the outer cycle of three iterations.

From these plots, it is seen that the reconstruction improves as the window size is increased, with the result of the previous set of iterations, plus an additive constant, as the initial guess.

The main limitations of this algorithm, are the following.

1. The rather ad-hoc addition of the small constant to the final reconstruction, to create the initial guess for the next set of iterations, may not always give an accurate reconstruction. It would be better to decide this constant adaptively, or from prior information about any extended structure of the source.

2. The window sizes chosen here, are specific for this particular source image. In reality, it would be better to be able to vary the window size adaptively,and arrive at one which gives the optimal solution.

In the graphs shown, it can be seen that for this particular source structure two outer iterations were sufficient to give a good approximation, and actually worsened because of the rather ad-hoc nature of the algorithm. This algorithm would benefit from being adaptive, along with a priori information.

3.2 Deconvolution by Singular Value Decomposition (SVD)

It was seen earlier that the convolution equation ($I^D = I^0 * B$) can be written in a matrix form, as a system of simultaneous linear equations as ($BM_{m\times n} I^0_{n\times 1} = I^D_{m\times 1}$)

To solve this set of equations,the beam matrix (BM) has to be inverted. But most often, the beam matrix is singular, and its inversion is not possible. The method of singular value decomposition, however, takes care of this singular nature of the beam matrix, by forming an approximation to its inverse. There may be infinite solutions (under-determined), or no exact solution at all (over-determined). In these cases, the SVD, by definition, searches among all possible solutions, to give the most appropriate one, from among the infinitely many that may satisfy the available equations. This will be the minimum norm solution, in the least squares sense.

With a complete flux window, The dirty image is itself the most appropriate solution (Principal solution) to this system of equations, since it represents a convolution of the actual image with the dirty beam. A convolution of the dirty image with the dirty beam, will also give the same dirty image.

Support information with a smaller flux window restricts the reconstruction to a limited interval, which can serve to bias the solution towards an image with limited extent, and away from the dirty image, which spans the entire data window.

This also serves to reduce the problem size. As the data and flux windows are imposed as constraints, the beam matrix accordingly reduces in dimension. The application of a data window, corresponds to discarding rows of the beam matrix, and a flux window will discard columns.

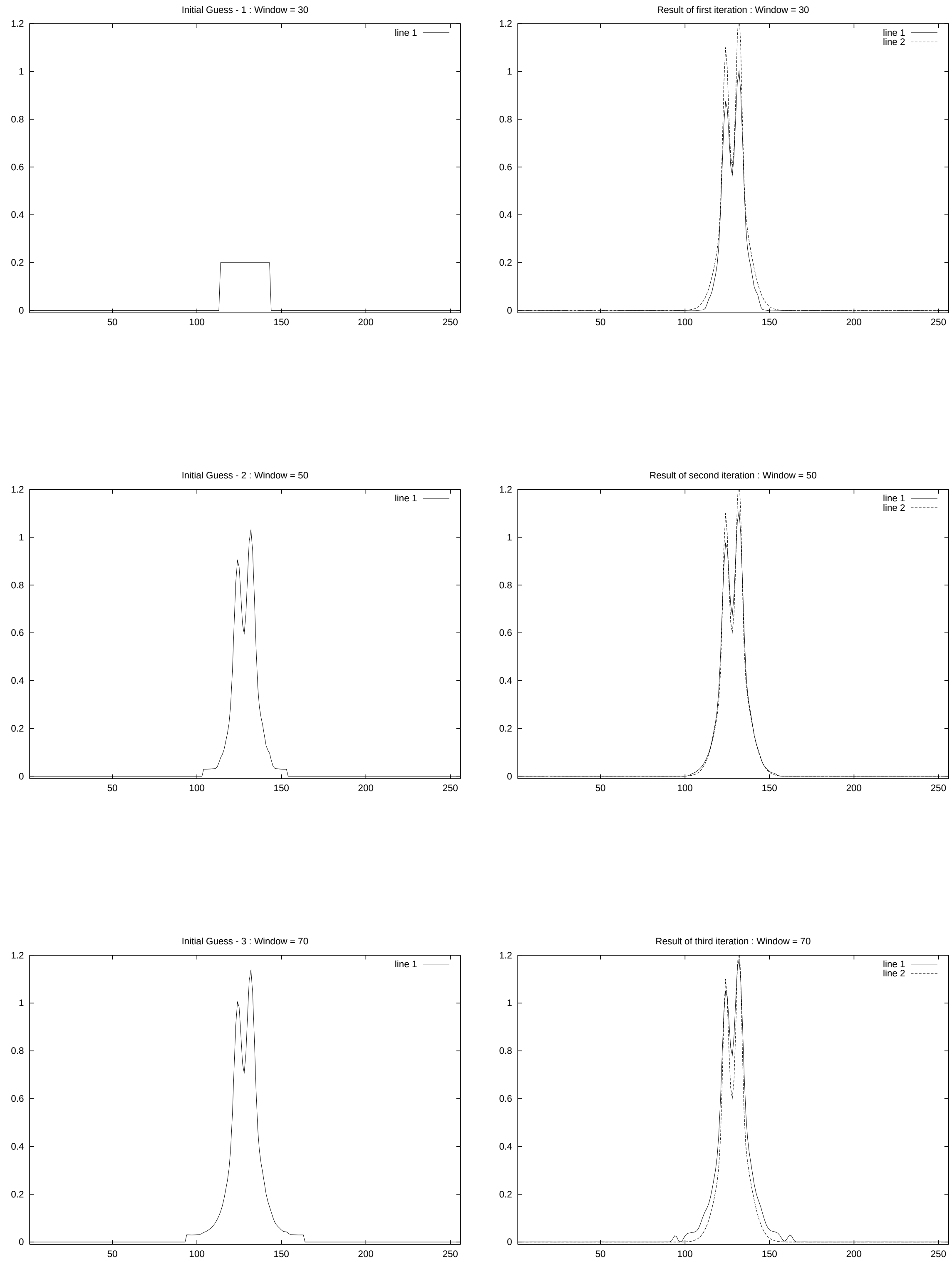


Figure 3: Plots of initial guesses and reconstructions at three stages

Therefore the SVD technique, with it's capability to produce the minimum norm solution, along with window constraints to regularize the problem, is a promising approach to solving the convolution equation, despite demanding computational requirements.

3.2.1 Singular value decomposition

SVD methods are based on the following theorem in linear algebra, that any matrix A of size $m \times n, (m >= n)$ can be expressed as a matrix product of three matrices as follows

$$A_{(m\times n)} = U_{(m\times n)} * W_{(n\times n)} * V^T_{(n\times n)} \tag{6}$$

where U and V are column orthogonal ($U^TU = VV^T = I$), and S is diagonal, with the elements representing the singular values.(When $m < n$, the matrix A is padded with zeros, to make $m = n$.)

The elements along the diagonal of W are singular values of A and the eigen values of A^TA, and are hence non negative.

Since U and V are orthogonal, their inverses are their transposes, and since W is diagonal, it's inverse is the diagonal matrix whose elements are the reciprocals of the elements w_j. The inverse of A can now be written as

$$A^{-1} = V_{(n\times n)} * [diag(1/w_j)] * U^T_{(n\times m)} \tag{7}$$

However, if any of the singular values are very small, or zero (i.e. A is singular) then this will not work. The condition number of a matrix is the ratio of the largest singular value, to the smallest one. A matrix is singular, if its condition number is infinite, and ill-conditioned, if its condition number is too large.

Consider the set of simultaneous equations in matrix form, as $A \times x = b$ where A is a square matrix and b and x are vectors. The equation defines A as a linear mapping from the vector space x to the vector space b. If A is singular, then there is some subspace of x, called the null space, that maps to zero. i.e.$Ax = 0$. The dimension of the null space is called the nullity of A. There will also exist a subset of b that A can reach. This subset of b is called the range of A, and its dimension is called the rank of A. If A is non singular, then the rank(A) is the total number of equations. If A is singular, then the rank will be less than the total number of equations.

The SVD explicitly constructs orthonormal bases for the range and null space of the matrix. The columns of U whose same numbered elements w_j are non-zero, are an orthonormal set of basis vectors that span the range. The columns of V whose same numbered elements are zero, are an orthonormal basis for the null space.

Hence,

$$x = V_{(n\times n)} * [diag(i/wj)] * [U^T_{(n\times m)}.b] \tag{8}$$

Now, if b is in the range of A, then even if A is singular, there exists a solution. In fact, there can be more than one solution, since any vector in the null space, can be added to x in any linear combination, and still give the same b, since these components only map to zero. This will be the solution vector of minimum length, and will minimize$|x|$.

If b is not in the range of A, then the SVD solution will be the closest to the ideal, in the least squares sense.

i.e. it minimizes $r = |Ax - b|$.

By the SVD method, in order to get the best possible approximation to the inverse of A, set all $(1/wj)$ to zero for all w_j less than a chosen cut-off.

This is possible, because the singular values being discarded, will correspond to rows of the beam matrix being discarded, which in turn will correspond to equations being discarded. Those equations would be the ones that would otherwise contribute to the instability of the solution. This can be seen in the case where there are nearly redundant equations in which case the true inverse will not exist. In this case, some of the diagonal elements of the W matrix will be very small, and which might make the solution unstable. Also, the columns of V corresponding to these small values, are linear combinations of x's, which are ill-determined, even by the supposedly over-determined set. These values and corresponding columns of V are then, insensitive to the data, and may also be a result of round-off errors.

When the inverse is to be calculated, the reciprocals of these small values become very large, and very significantly distort the solution vector. Zeroing these very large reciprocals, will stabilize the solution. It will also reduce the number of free parameters to determine.

Therefore this method is especially useful in cases when the matrix A is both over-determined (number of data points is greater than the number of parameters) and under-determined (ambiguous combinations of parameters exist).

3.2.2 Solution by Singular Value Decomposition

The standard linear least squares method to solve a set of simultaneous equations, given by

$$y(x) = \sum_{k=1}^{M} a_k X_k(x) \tag{9}$$

is controlled by minimizing the parameter

$$\chi^2 = \sum_{i=1}^{N} \left[\frac{y_i - \sum_{k=1}^{M} a_k X_k(x_i)}{\sigma_i} \right]^2 \tag{10}$$

So, differentiating χ^2 w.r.to a_k,

$$\sum_{i=1}^{N} \frac{1}{\sigma_i^2} \left[y_i - \sum_{j=1}^{M} a_j X_j(x_i) \right] X_k(x_i) = 0 \quad \text{k} = 1, ..., \text{M} \tag{11}$$

or

$$\sum_{i=1}^{M} \alpha_{kj} a_j = \beta_k \tag{12}$$

where

$$\alpha_{kj} = \sum_{i=1}^{N} \frac{X_j(x_i) X_k(x_i)}{\sigma_i^2} \quad [\alpha] = A^T A \tag{13}$$

and

$$\beta_k = \sum_{i=1}^{N} \frac{y_i X_k(x_i)}{\sigma_i^2} \quad [\beta] = A^T b \tag{14}$$

These are called the normal equations, which in matrix form, translate to

$$(A^T A) a = A^T b \tag{15}$$

The parameter to be minimized here, χ^2, in matrix form would be written as $\chi^2 = |Aa - b|^2$ where a is the solution vector.

This is the exactly what the SVD by definition, minimizes, while searching for the most appropriate solution. Therefore this least square χ^2 minimization can be carried out just by using the SVD method to find a minimum norm solution.

The minimum norm solution vector will then be given by

$$a = \sum_{i=1}^{M} \left(\frac{U_{(i)} . b}{w_i} \right) V_{(i)} \tag{16}$$

where $U(i)$ and $V(i)$ represent the columns of U and V respectively, and w_i are the singular values.

3.2.3 Algorithm

For particular data and flux windows, the following steps are followed.

1. The full beam matrix is truncated according to the specified windows. Only the rows corresponding to the positions in the data window,are retained, and similarly, only those columns corresponding to the flux window, are retained.

2. The singular value decomposition of the truncated beam matrix is evaluated. $A = U * W * V^T$

3. The inverse of the singular value matrix is first calculated, by finding the reciprocals of the diagonal elements. Here, for very small (less than a tolerance threshold),or zero singular values, the reciprocals, which would be very large, or infinite, are set to zero.

4. The inverse of the truncated beam matrix is evaluated as $A^{-1} = V * [W^{-1}] * U^T$

5. The solution vector is calculated from $x = A^{-1} * b$

6. This solution is then convolved with the clean beam, and the mean square deviations from the ideal solution, in both the spatial, and visibility domains, are calculated.

3.2.4 Implementation and Results

This implementation uses a complete data window, and varies only the flux window. Hence, the truncated beam matrix always has 256 rows, whereas the number of columns is dictated by the width of the flux window.Figure 3.2.4 shows the beam matrix and its SVD.

The tolerance threshold is defined as half the reciprocal of the window size. This chooses an appropriate number of singular values, for the reconstruction.

Initially, the SVD was performed on truncated beam matrices, with a complete data window, and varying flux windows, and the singular values obtained, were plotted against the order index (Figure 3.2.4).

It is seen from this plot, that as the window size is reduced, the ratio of the number of significant singular values to the total number of singular values, increases.

Therefore, for smaller window sizes, the last few singular values, although significant in magnitude, may still contribute to the irregularities in the data. Hence it may be better to use a smaller fraction of the significant values, for smaller window sizes.

This has been implemented by setting the tolerance threshold for the singular values,as half the reciprocal of the window size. For smaller windows, this results in a higher threshold, and therefore fewer chosen significant singular values.

This was found to bring in stability to the reconstructions at smaller window sizes. For larger window sizes, it did not make much of a difference, since the threshold would just vary in the steepest parts of the curve, and all the significant singular values would always be above it.

Program runs were taken for varying window sizes, with and without this adaptive thresholding, and the plots of the mean square difference in the image as well as visibility domains, with respect to window size, are as in Figure 3.2.4.

It is seen from the above plots that this kind of adaptive thresholding can be effective in maintaining smoothness in the reconstructions, and consistency, with different window sizes. Also, even with windows smaller than the extent of the source, an acceptable fit to that region of the image, is obtained, as long as the window is larger than around one third of the extent of the source. However, this is a rather intuitive way of adaptively defining the threshold, and a more concrete, analytical method of defining it, would be better.

From the plots it is also seen that there is a 'optimal' region (roughly from window size 34 to 48) where the mean square differences in both, the image and visibility domains, are minimum. Looking at these plots, it is seen that, beginning with a window size of 20, as the window size increases, the mean square differences

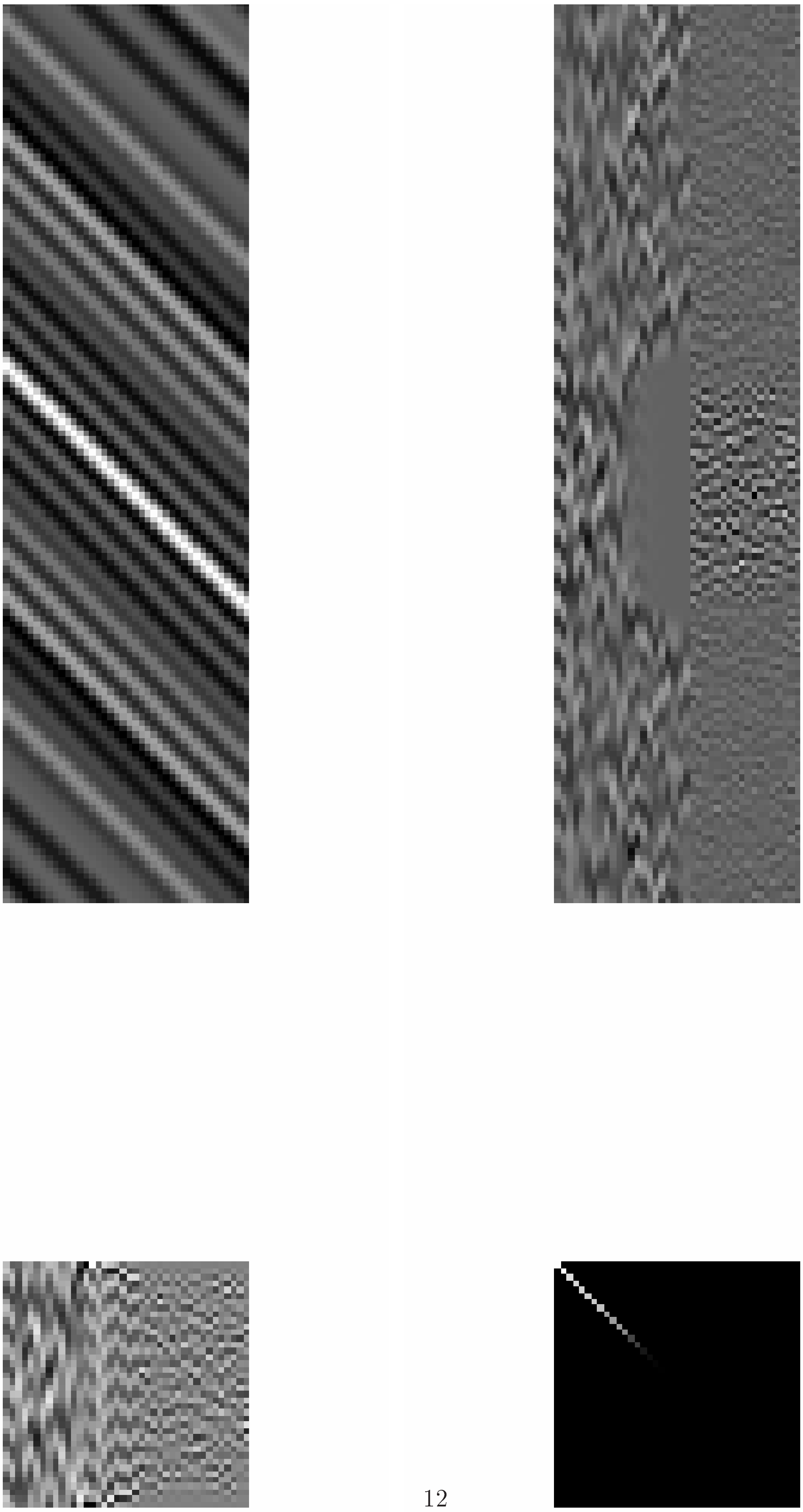


Figure 4: Beam Matrix and its Singular Value Decomposition : U,V and S

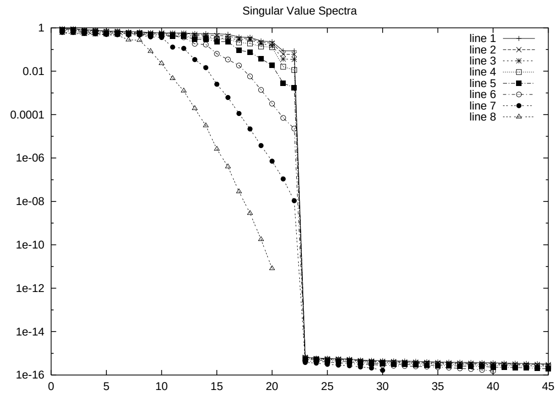
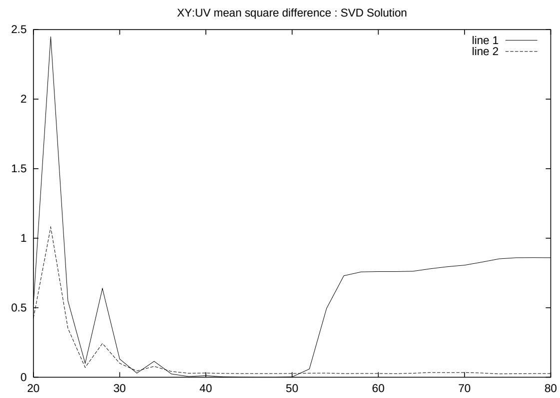


Figure 5: Singular Value Spectra



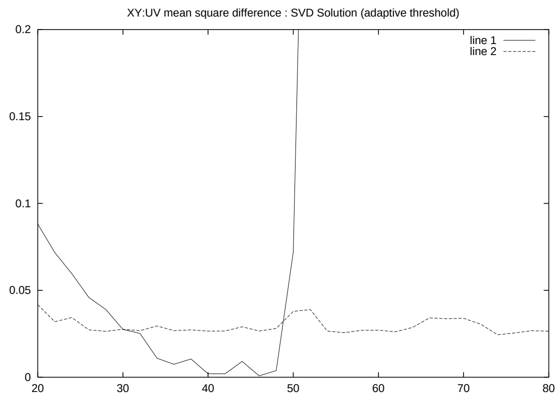


Figure 6: Plot of XY-UV mean square difference, for varying window sizes

in the image as well as visibility domains, first decrease. After a window size of around 50, the image domain mean square differences rise sharply, and steadily keep increasing.

This can be explained as follows. The actual source image in the simulation model, spans around 45 values, out of 256. Below a window size of 35, the flux window cuts into the image, and a reconstruction is obtained only within that region. The mean square differences in both domains,then decrease, as the window size is increased to approach the actual source extent, and they reach their minimum in this region. For the mean square difference in the visibility domain, this means that the measured data is being fitted almost exactly and for the mean square difference in the image domain, it is an indication of the actual source extent.

Beyond a window size of 48, the curve for the image domain differences, first rises sharply and then starts steadily increasing. The reason for this is that as the window size is increased beyond the actual extent of the source, the reconstruction starts moving towards the dirty image, and the reconstruction becomes worse.(For a full flux window, the reconstruction will be the dirty image itself, the Principal solution of the convolution equation).

The mean square differences in the visibility domain, however, remain more

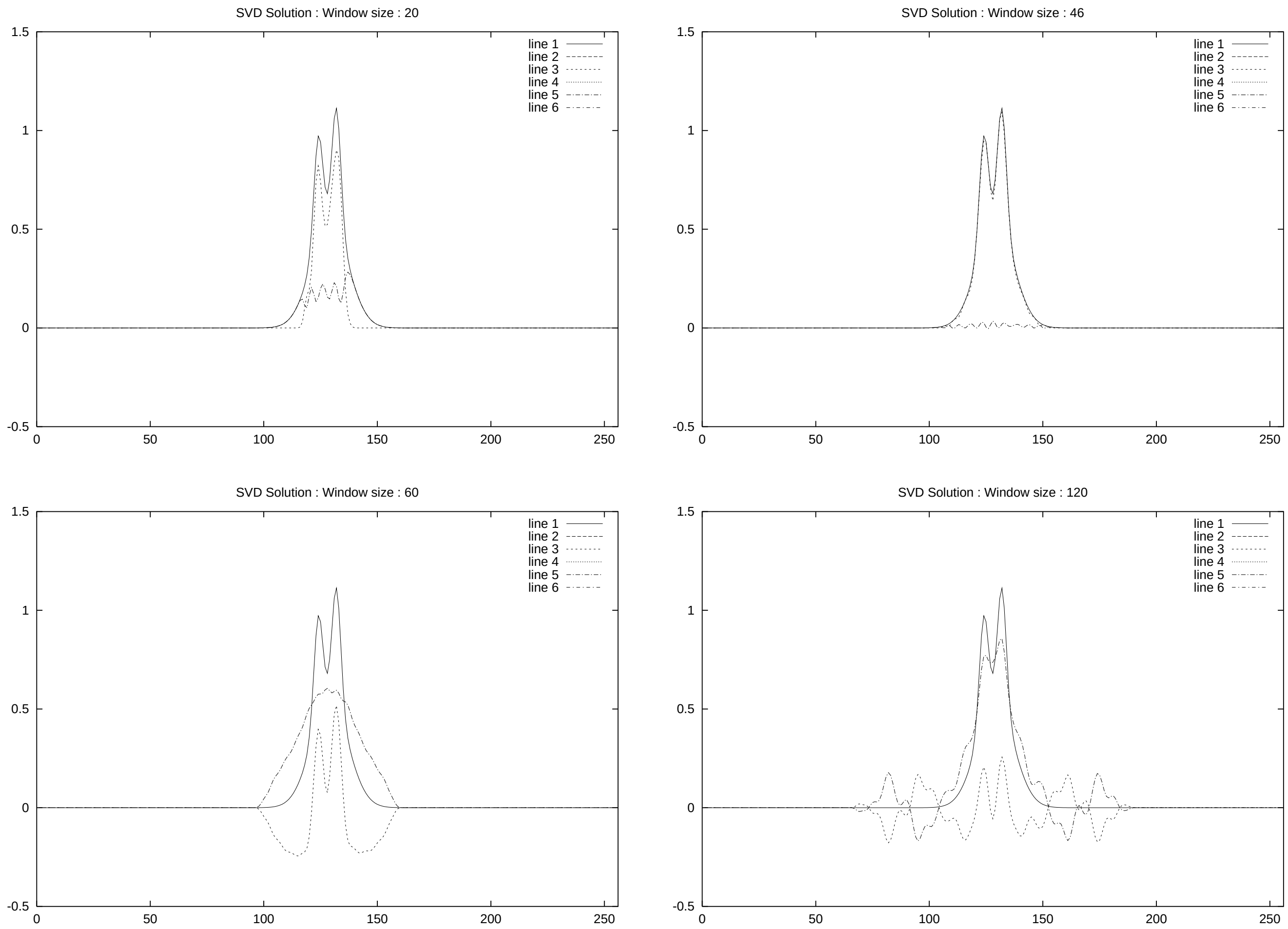


Figure 7: SVD Deconvolution - plots of reconstructions

or less at the level attained around the optimal window size, indicating that the visibilities at the measured points have been fitted as accurately as possible, and now, only the unmeasured values are varying.

As long as the window size remains less than the actual source extent, it is seen that the reconstruction is predominantly positive. But, as soon as the window crosses the actual source size, and the reconstruction starts following the dirty image, a negative 'bowl' appears. This can be attributed to the missing zero spacing values, which would have otherwise contributed to reconstructing extended source structure.

It is then possible to conclude that with smaller flux windows, higher spatial frequencies (small scale features) dominate the reconstruction, and as the window size is increased, lower spatial frequency components (large scale, or extended features) get included.

The following are plots of reconstructions at a few particular window sizes in which the features mentioned above, are apparent (Figure 3.2.4).

It is thus seen that the SVD deconvolution, for specific data and flux windows, produces a minimum norm solution.

The accuracy of the reconstruction a function of the window size, and of the number of significant singular values chosen to calculate the approximation to the inverse of the truncated beam matrix.

With smaller window sizes, the choice of a threshold for selecting singular values becomes important.

At large window sizes, negative flux appears because of the missing zero spacing values, and the fact that the minimum norm solution starts getting biased towards the dirty image itself.

The SVD and subsequent matrix multiplication are computationally expensive operations, and will considerably slow down the iterations, at large window sizes.

A small amount of additive gaussian noise, was added and the SVD results did not vary much.

3.3 Deconvolution by Non Negative Least Squares (NNLS)

We have seen that the basic convolution equation $(B * I^0 = I^D)$, can be written in matrix form, as $(BMI^0 = I^D)$

In neither of the methods described earlier, has a positivity constraint been applied analytically to the problem.

The method of Non Negative Least Squares, applies to those least square problems with linear inequality constraints. The general representation is as follows.

Minimize $||Ax - b||$ subject to $Gx >= h$

where $Ax = b$ is the set of simultaneous linear equations to solve, and in this case $A_{m\times n}$ is the beam matrix, and b is the dirty image. $G_{m\times n}$ is a constraint matrix, and $h_{m\times 1}$ is the element constraint vector. In the case of the positivity constraint, G is the identity matrix, and h is identically zero.

There can be multiple solutions to this problem, all satisfying the constraint equations. Out of all these possible solutions, the algorithm must pick the one closest to the ideal solution.

3.3.1 Non Negative Least Squares

An intuitive understanding of the technique is as follows.

The constraint equations define a subset of the solution space. Each inequality constraint divides the entire space into half, and their intersection defines the set of solutions that satisfy the inequality constraints. In the case of positivity as the constraint, in three dimensional Cartesian space with the basis vectors along the co-ordinate axes, the solution space is confined to the positive quadrant, including the co-ordinate surfaces.

Take a function (in this case the norm $||Ax - b||$), which defines a surface, and which has a minima at a certain point. The negative gradient vector of the function, evaluated at any point on the surface, will point towards the local minimum (which may also be the global minimum). At the point of minima, the gradient will have a value zero.

If the direct solution to the system of equations, lies within the constrained solution space, then the obtained solution is exact. But, if the actual solution lies outside the constrained solution space, then obtained solution will be the point within that space, nearest to the actual solution. The negative gradient, evaluated at this chosen point, will point in the direction of the actual solution, with its

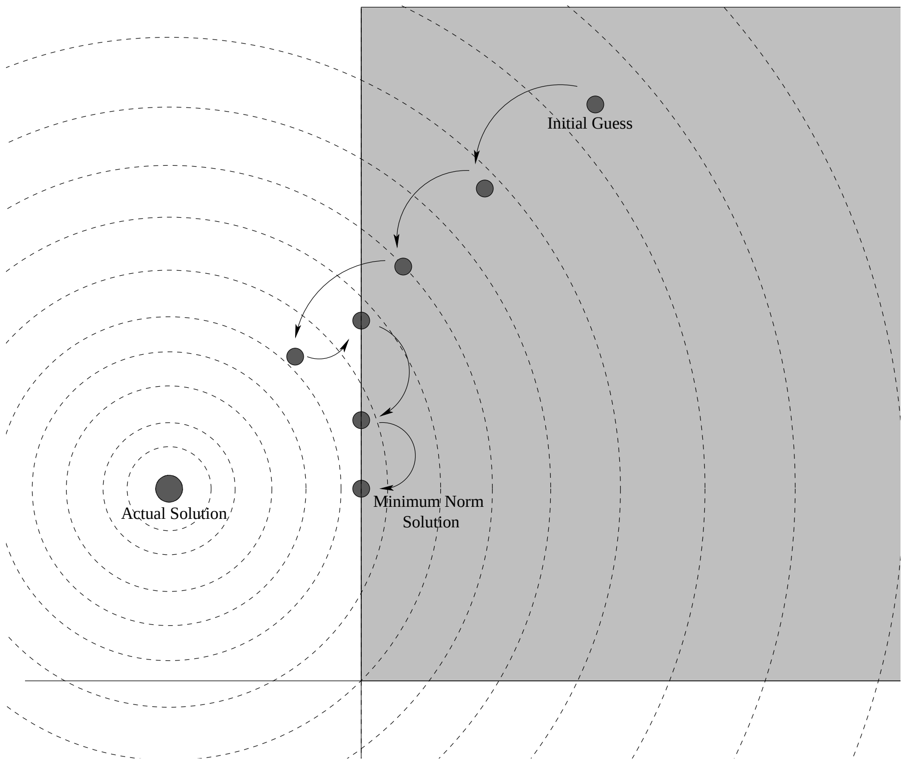


Figure 8: Path of the solution in the parameter space

magnitude given by the distance from it to the actual solution. The algorithm ensures that this magnitude is the minimal.

The algorithm takes an initial guess, and calculates the negative gradient of the norm function at that point. The algorithm then moves the solution along the axis corresponding to the component with the maximum coefficient. At each iteration, this amounts to moving the solution, along the gradient vector, closer to the minimum of the function, as fast as possible. It terminates when all the component coefficients in the gradient vector are negative, which means that, the only way the solution can vary, to move closer to the actual solution, is out of the constrained solution space.

The algorithm begins with all component coefficients equal to zero, and slowly builds up a set of component coefficients that are fits to the parameters defining the component axes.

A pictorial representation of the working of this algorithm, in a two dimensional space, with the norm function as a contour plot, is given in Figure 8.

The solution moves towards the actual solution, until it reaches a constraint surface, after which, it starts moving on that surface, to get to the point closest to the actual solution.

3.3.2 Algorithm

The NNLS algorithm holds two sets of indices Z and P, to indicate whether a given tentative solution element is zero (i.e. whether that parameter has been limited by the constraint surface and cannot decrease below a value of zero) or not.

Z is a list of indices corresponding to the set of parameters (axis components) which have zero coefficient.

P is a list of indices corresponding to the set of parameters which can take arbitrary values.

x is the solution vector that gets constructed during the iterations. This corresponds to the flux window.

The primary loop transfers indices from Z to P, to construct a possible solution vector, by fitting the parameters corresponding to the indices in P.

The secondary loop solves the deconvolution equation with a truncated beam matrix, which calculates a solution using only the parameters currently present in the set P.

If all components of the so obtained partial solution vector are positive, then this is taken as the initial guess for the next iteration.

If any of the components of the solution vector are negative, it means that the algorithm has moved the solution out of the allowed constraint space, and some parameters have become negative. An interpolation is done between this solution and the previous one, to obtain a solution on a constraint surface, to maintain positivity.In the result, any parameter having the value zero, is moved from P to Z.

1. All elements of the solution vector x are set to zero, all solution component indices are put into Z, and P is kept empty.

2. The primary loop begins here. The gradient vector w is computed from the current value of x, as $w = A^T(f - Ax)$.

3. If Z is empty, or if all elements of w with indices in Z have values $<= 0$, a solution has been obtained and the iteration terminates.

   Z being empty, corresponds to a solution where all parameters have been fit and have non zero values. This means that the actual solution is within the constrained solution space. If Z is not empty, it means that some parameters have been forced to take the value zero, either because they actually do not affect the solution, or since they have reached the constraint surface, and are not allowed to go further. Therefore all corresponding elements of w being negative, will mean that the only way the solution can move, is out of the allowed solution space, which means that the minimum norm solution has been reached.

4. If any value in w corresponding to an index in Z is positive, it means that a parameter that has not yet been incorporated in the solution can be used to move the solution vector within the solution space, to get closer to the actual solution. Therefore the parameter corresponding to the maximum of such w values is taken into the solution, and its index is moved from Z to P.

5. The secondary loop begins here, and the matrix equation $A_{pos}z = f$ is solved for z, where A_{pos} is A with columns corresponding to indices in Z, set to zero. This means that the solution obtained will be in terms of only those parameters that are currently being included in the solution, i.e. the parameters corresponding to indices in the set P.

6. If all the values of z with indices corresponding to those in P, are > 0, then this is an acceptable new solution. z is copied to x and the main loop is continued at (2).

7. If some of the values in z are $<= 0$, then only a fraction of z can be taken as a trial solution. z and the old trial solution x, have to be combined in a way that there are no negative valued components in the trial solution vector. This is an interpolation in which, to each value in x, is added a fraction of the corresponding value in z, such that all the parameters are now positive. This fraction is chosen such that the most negative value in z with respect to its corresponding value in x, when multiplied by the fraction, and added to that value in x, gives zero.

   The fraction is found as the minimum of $\alpha = x_q/(x_q - z_q)$ for all q.

   The linear sum $x = x + \alpha(z - x)$ is formed.

8. At least one value in x now has the value zero, which means that those parameters are not required in the solution, and can be removed. Hence all indices for which the corresponding element in x is zero, are moved from P to Z. This is like removing dependent variables from the solution.

   The secondary loop is now continued from (5).

3.3.3 Implementation and Results

The data window was taken as the entire data set, and the flux window was varied. The convolution matrix equation was solved in the secondary loop via a QR decomposition.

The algorithm was run for flux window sizes varying from 20 to 80, and the following plots of mean square differences in both, the image as well as the visibility domain, with respect to window size were obtained (Figure 3.3.3).

It can be seen that as long as the flux window is equal to or more than the actual source extent, the window size does not make much of a difference, to either the fit in the image domain, or to the visibility values.

The following reconstructions were obtained with flux window sizes of 20 and 50, where the actual source extent is around 40 values (Figure 3.3.3).

There is a bug in the implementation of this algorithm, and it is found to almost never to go to convergence. To counter this, the terminating condition has been chosen to be when the difference between solution vectors in successive iterations, stop varying.

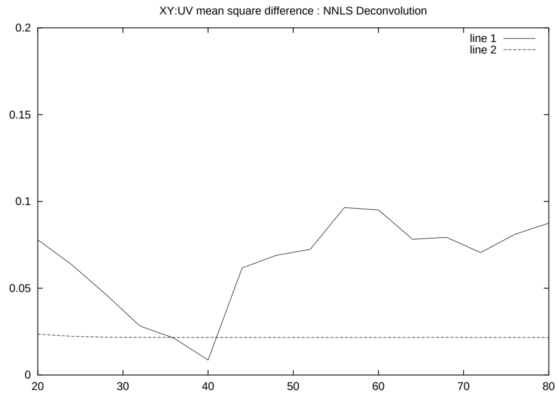


Figure 9: XY-UV mean square differences with varying window size

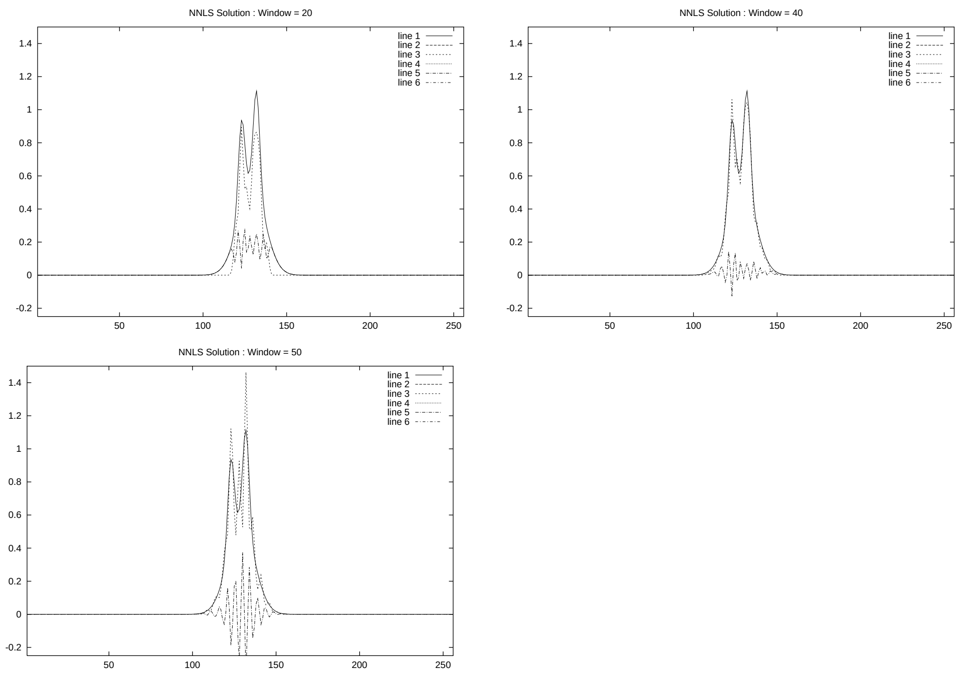


Figure 10: NNLS reconstructions

Also, a QR decomposition was used to solve the deconvolution equation in the intermediate iterations. in this implementation, a direct QR decomposition was used in each iteration. It is however possible, to update the decomposition matrices dynamically, in successive iterations, to prevent a complete re-computation at each stage.

4 Conclusion and further work

The three algorithms were studied and implemented for the one dimensional case. The effect of varying flux window size on the accuracy of the reconstructions, was studied and the following basic interpretations were made.

The iterative algorithm, was implemented, mainly to do a comparison, with the other two algebraic deconvolution methods. The iterative algorithm involved a priori information, with which accurate reconstructions were obtained.

1. The existence of multiple solutions, was seen in the observation that an accurate fit in the visibility domain does not guarantee a reconstruction very close to the original.

2. The reconstruction depends on the window size

3. Effect of the missing short spacings, is apparent

With the SVD algorithm, small windows (smaller than the actual source extent) gave good fits within the flux window, if the appropriate number of singular values were chosen for the reconstruction. The optimal fits were obtained with windows approximately equal to the actual source extent. As the window size was increased, a negative 'bowl' appeared which could be attributed to the missing short spacings information. As the window size was increased further, the reconstruction approached the Principal solution. No positivity constraint was applied. The reconstruction was controlled only with the flux window, and the inherent property of the SVD to give a minimum norm solution.

1. There is an optimal window size, which can be related to the number of significant singular values chosen for the reconstruction.

2. At small window sizes, only small scale details get reconstructed, while at larger window sizes, large scale information gets included.

3. It is sensitive to missing short spacings

With the NNLS algorithm, as long as the flux window was larger than the source extent, the reconstructions did not depend much on the window size. However, extended structure in the source was not accurately reconstructed. The algorithm fitted only a subset of the parameters, and this needed a convolution with the clean beam, to smooth out. But positivity is always satisfied in an NNLS reconstruction.

Some ideas that may be worth pursuing, are as follows.

1. In the SVD deconvolution, the accuracy of the reconstruction depends on the number of singular values included as well as the window size. An analytical method of deciding the required singular values for a given window size, could lead to dynamically varying the window size, to reach the optimum.

2. The U and V matrices contain scale information about the beam matrix and the solution. A more specific technique to use this feature of the SVD, could be possible.

3. Optimization - The large computations involved in the SVD, can be optimized using the symmetry properties of matrices, for example, the SVD of a circulant - dft of top row of the beam matrix.

4. All the comparisons were done with the data window fixed to the entire dirty image. Effects of changing the data window should be studied.

5. SVD and Image Compression An image, considered as a matrix, can be decomposed into the U,S and V matrices by the SVD method. It can be shown that the same image can be accurately represented by a back reconstruction, using only the significant singular values. This reduces the number of values required to represent the entire image, from N^2 to $(2N+1)*ns$, where ns is the number of significant singular values chosen. The deconvolution problem, expressed in terms of these limited parameters, would result in a greatly reduced number of free parameters in the system. It would also achieve compression in the storage and manipulation of the images.

6. In the SVD Deconvolution, the data window was the dirty image. Now, this can be considered to be a highly correlated system of values, since they have all been derived from just over 20 Fourier components. A large reduction in the number of parameters involved in the problem, may be achieved, if the data window could be considered to be the measured visibility values themselves. The problem could then be formulated by defining the beam matrix separately in terms of the Fourier transform matrix and the visibility sampling function.

5 References

1. Press et al,'Numerical Recipes in C',1991,Cambridge University Press.

2. Daniel Briggs,'High fidelity Deconvolution of Moderately Resolved Sources',1995.