NCRA Technical Note - December 2008

Identifying baseline problems from UV tracks

N. G. Kantharia, Rajaram Nityananda
National Centre for Radio Astrophysics, TIFR, Pune

1 Motivation

The Giant Metrewave Radio Telescope operates at low radio frequencies where several man-made problems such as narrow-band radio frequency interference (RFI), wide-band RFI due to power line discharge etc, corrupt the data. Moreover generic problems such as cross-talk between antennas and malfunctioning baselines also affect the data. All the above are additive errors which affect the final image as follows:

$$Visibilities + error \longleftrightarrow Image + FT(error); \qquad (1)$$

FT(error) is the Fourier transform of the error. Basically the image has an extra component added to it. For example, RFI which persists throughout an observing run and is not excised will generate a wide-angle pattern in the final image with a fringe rate equal to that of the baseline which picked it up added to the sky image. If bad baselines persist through an observing run, they will generate a different response in the image plane. If the bad data generates a discernible response in the image plane then it is imperative to remove it in order to improve the S/N ratio of the image.

If we can identify the data which cause the above problems then that data can be edited and the final image would have a better dynamic range. In the recent past, we have received reports regarding a weak ring-like structure centred close to the phase centre in the final deconvolved image from several GMRT users. The pattern is insensitive to self-calibration or deeper deconvolution. Since this problem is likely caused by bad baselines (Dave Green, private communication, R. D. Ekers, in Synthesis Imaging in Radio Astronomy II, ASP Conference Series, Vol. 180, 1999, Ed: G. B. Taylor, C. L. Carilli, R. A. Perley), we have developed a procedure, which starts from the artifacts in the residual map, to assist the GMRT user in identifying the bad baselines. In this note, we outline the algorithm, its implementation and show the results we have obtained. We believe this is a convenient way to identify bad baselines from corrupted data which complements several existing methods/programmes/tasks in AIPS such as UVFLG, TVFLG, SPFLG etc.

2 Principle of identifying antenna pairs from uv tracks

Fourier transforming artifacts in the residual sky image generates a UV image clearly showing tracks of the general shape expected from the source and telescope geometry (see next section for details). Each baseline clearly sweeps out a cone (see A'B' in Fig. 1), and the track is the projection of this onto a plane normal to the line of sight (LOS) from the source to the earth (Fig. 1). Only part of this is of course visible.

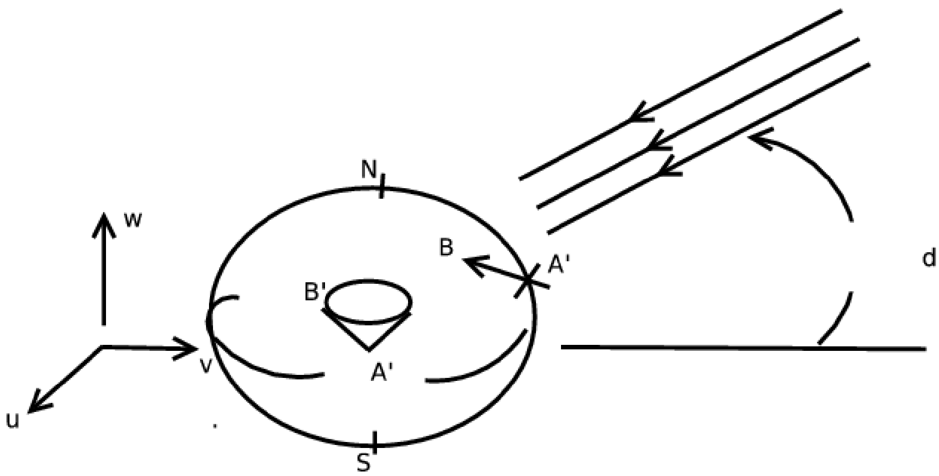


Figure 1: The baseline AB made by two antennas placed at A and B on the earth's surface has been moved to the earth's centre as A′B′ & B′ sweeps out a circle as the earth rotates. Notice that the baseline describes a cone and the UV track is the projection on a plane normal to the LOS.

For convenience, let us use a coordinate system with (U,V) lying in the plane of the equator and W along the polar axis (see Fig 1). The circle described by B′ has the equation $W = P$ and $U^2 + V^2 = E^2$, where P stands for the polar component of the baseline and E the equatorial component. These don't change with time, but (in a space fixed coordinate system) U & V do, as $U = E\cos(H-H_0)$, $V = E\sin(H-H_0)$, H being the hour angle. If we choose the V axis to point to the RA of the source, then a vector along V appears largest in projection (on to the plane transverse to the earth-source line) for $\delta = \frac{\pi}{2}$ while the W component is maximum when viewed from the equator, $\delta = 0$. The projected baseline track is therefore given by an ellipse centred on $(0, P\cos\delta)$ (see Fig. 2) with major axis, along U, of length E, and minor axis, along V, of length $E\sin\delta$ (u & v are projected values, and equal to U & V only for & $\delta = \pi/2$, i.e.polar viewing). Thus, the equation to the elliptic track is

$$\frac{(v - P\cos\delta)^2}{E^2 sin^2\delta} + \frac{u^2}{E^2} = 1$$

Given two points (u_1, v_1) and (u_2, v_2) on this curve,

$$(v_1 - P\cos\delta)^2 + u_1^2\sin^2\delta = (v_2 - P\cos^\delta)^2 + u_2^2\sin^2\delta = E^2$$

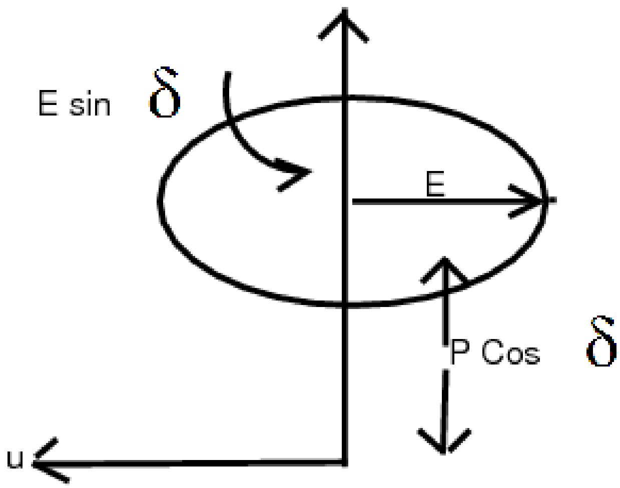


Figure 2: The figure shows the equatorial (E) and polar (P) components of the uv track which can be estimated from the uv track and can in turn be used to uniquely identify the antennas involved in the baseline.

which immediately gives

$$2P\cos\delta \quad (v_2 - v_1) = (v_2^2 - v_1^2) + \sin^2\delta \quad (u_2^2 - u_1^2)$$

From the above,

$$P = \frac{(v_2^2 - v_1^2) + \sin^2\delta \quad (u_2^2 - u_1^2)}{2cos\delta \quad (v_2 - v_1)}$$

and

$$E^2 = \frac{(v - P\cos\delta)^2}{sin^2\delta} + u^2$$

Substituting the value of P gives E^2.

Since the equatorial (E) and polar (P) coordinates which are properties of a baseline and independent of time, the above exercise allows one to estimate these coordinates from the antenna positions and compare with those estimated from the uv track of a bad baseline. This exercise thus helps identify the bad baseline which can then be removed and the image quality improved. We describe the implementation and results from this procedure below. However since there is a finite possibility of the baseline being wrongly identified especially in those parts of the uv space where there is crowding and hence the E, P coordinates of more than one baseline are likely to be similar. To check this, we also made a list of baselines which are likely to be misidentified. We find about 57 baselines out of a total 435 baselines which are likely to be confusing because the difference in their equatorial and polar coordinates is within 50m of another baseline. Most of these, as expected, are central square baselines.

3 Implementation and Results

We have developed a programme in C which uses the above algorithm and identifies the baseline given coordinates of two separate points on the UV track. FFT of the source-subtracted residual image results in an uv image which shows the uv tracks and the real and imaginary parts of the visibility amplitude. Since the source response has been removed, the residual image are expected to contain only artifacts and hence any signature in the uv image can safely be assumed to be due to bad data.

The data obtained in the raw *lta* data format were converted into FITS using *listscan* and *gvfits*. The FITS data were then imported to NRAO AIPS [1]. and the data calibrated and imaged. Self-calibration of the data was also effected. The image showed the presence of ring-like artifacts originating near the pointing centre and mimicking a pseudo point source. We subtracted the clean components of all the sources from the uv data (UVSUB) and then imaged the residual uv data. The concentric rings persisted, were the major features in the image and are indeed the artifacts which we are trying to remove.

Fig. 3 shows the image of a field made after the sources in the field were subtracted from the uv data. The concentric ring-like structure centred close to the pointing centre is due to a bad baseline and needs to be removed before the dynamic range of the image can improve. Both RR and LL show the ring-like structure.

We used the task *FFT* in AIPS on the images shown in Fig. 3. Since the task was used on the sky map, we obtained a complex image of the uv plane. In Fig. 4, we show the real part of the FFT in the uv plane. The axes are labelled in u and v kλ. uv tracks made by various GMRT baselines as the earth rotates can be seen for both RR and LL. Since the sources have been removed, all the tracks should ideally register only noise. However the tracks pointed at by the arrows in Fig. 4 are brighter as compared to rest of the tracks and stand out. These are the bad baselines that probably persisted throughout the observations and gave rise to the concentric ring pattern seen in Fig. 3. We used the programme we have developed based on the above algorithm, *badbase* to identify this bad data.

The first task was to obtain the uv coordinates of any two points on the baseline track using which the programme *badbase* would find the closest match with numbers estimated from the measured values of the antenna positions which are generally referred to as B_x, B_y and B_z metres. We used a couple of different ways within AIPS involving different verbs and tasks to find these coordinates. Since many close baselines can have overlapping uv tracks, we stress the need for accurately finding two points lying on the same track. If by mistake the two points are from two different tracks, *badbase* will not be able to give a good match of the uv track to a possible baseline. In Appendix A, we suggest a couple of ways in AIPS using which the coordinates can be obtained.

[1]The National Radio Astronomy Observatory is a facility of the National Science Foundation operated under cooperative agreement by Associated Universities, Inc.

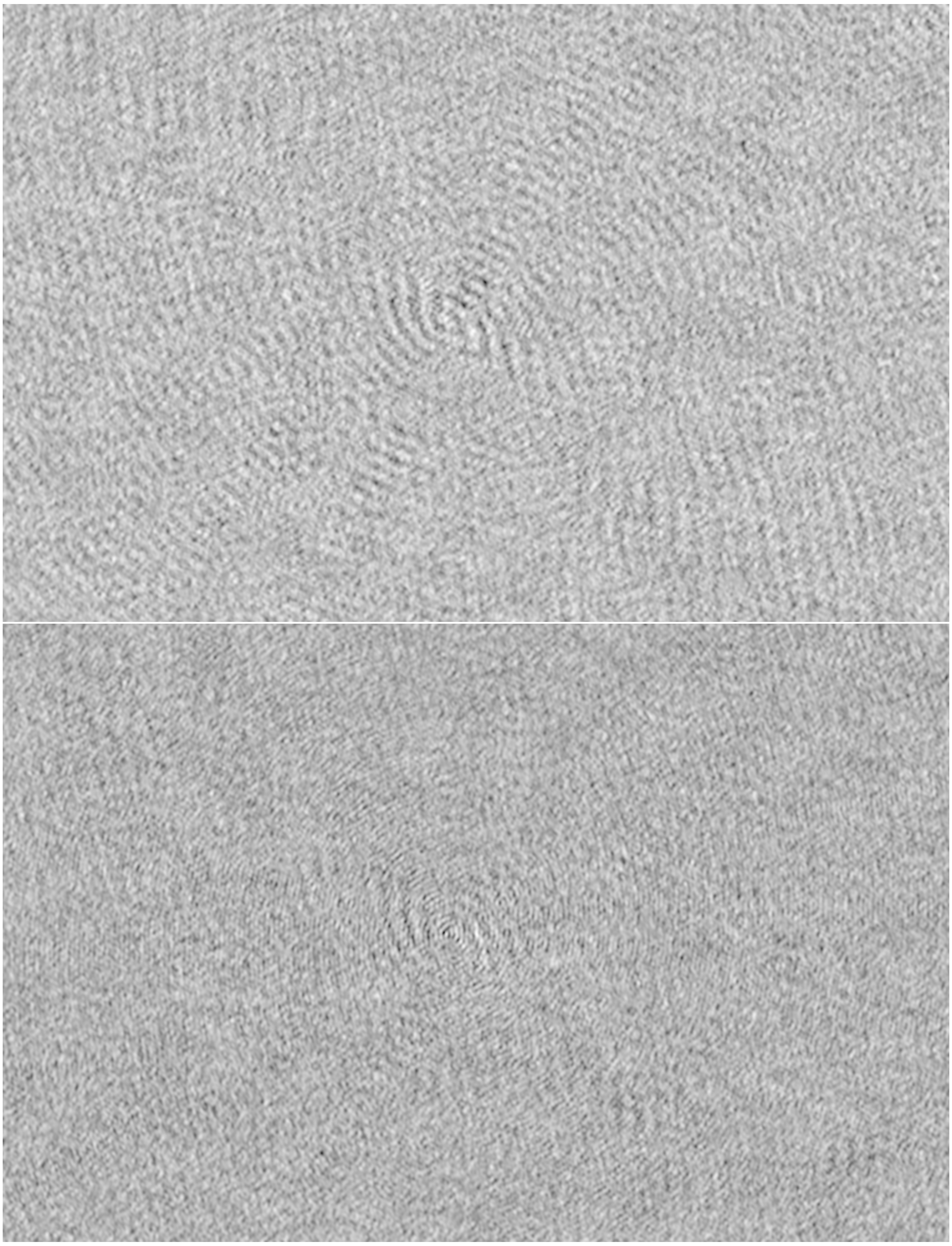


Figure 3: The 1.4 GHz RR (top) and LL (bottom) images of the central part of the primary beam. Notice the concentric ring-like structure near the centre of the image. No source is responsible for the structure which at first glance looks like a dirty beam. The dynamic range of the image is limited by the systematics. Moreover note that the fringe frequency of the dominant ring-like structure in the LL is larger than that in RR.

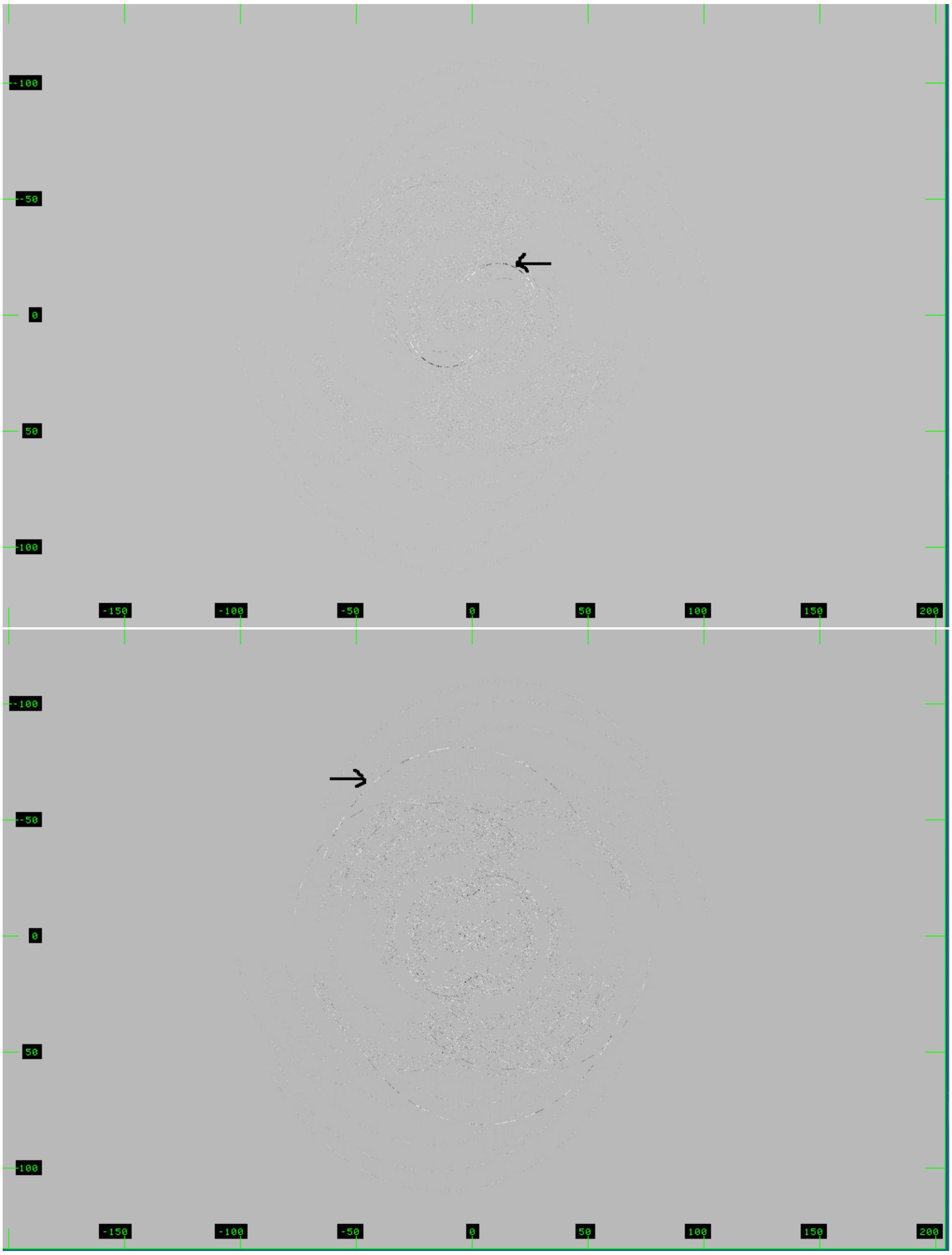


Figure 4: The FFT of the data shown in Fig. 3. RR (top) and LL (bottom) images of the FFT are shown. The UV tracks which give rise to the dominant ring-like structure in the RR and LL images are shown by an arrow. Our algorithm helps identify such bad UV tracks. The X (VV) and Y (UU) axes are labelled in kλ.

Once the coordinates of two points have been obtained, these are to be given to *badbase* alongwith the declination of the source in degrees, wavelength of observation in cm and the tolerance in metres upto which you would like the programme to probe for a match. We have been able to find matches to within 50m of a probable baseline and in many cases the match is within 20m. On the other hand, we would also like to warn users that in crowded regions of the uv plane there are chances of misidentifying baselines since there are likely to be more than one baseline within a given tolerance.

Once we identified the corrupt baselines using *badbase*, these were removed from the uv data and images were made again as shown in Fig. 5. The strong ring-like feature near the phase centre is not present in these images although the Stokes RR seems to show a weak ring-like structure which is the result of another bad baseline. The programme can be used iteratively. However, with each iteration, caution should be exercised to avoid good baselines being edited. One way which would work is to look at a small range of extremum values in the image and check for any bad baselines which appears to show 'signal'. If part of a baseline is bad then only that part should be removed. We suggest that one does some extra check on the baseline before editing it.

4 Summary

We have described an algorithm which works from artifacts in the image plane to identify bad baselines which result in ring-like artifacts in the image plane and limit the dynamic range at rms noise levels less than half a mJy. The implementation involves identifying two distinct points on the uv track and finding their uv coordinates. These coordinates have to be given to *badbase*, the programme that we have developed which compares the equatorial and polar coordinates obtained from the uv track with those estimated from the antenna positions and gives the closest match. Removing this baseline generally results in the ring-like artifacts disappearing from the image. We also note that this is one more way of identifying bad data in addition to several like UVFLG, TVFLG, SPFLG etc which exist in the AIPS environment.

5 Acknowledgements

We thank Ramya Sethuram of IIA for providing us with calibrated radio data which had the problem of concentric ring-like structure near the phase centre and also using our procedure several times helping us gain confidence in it appreciably. We thank Reena Shrikumar and Annabhat Joshi for help in preparing this document.

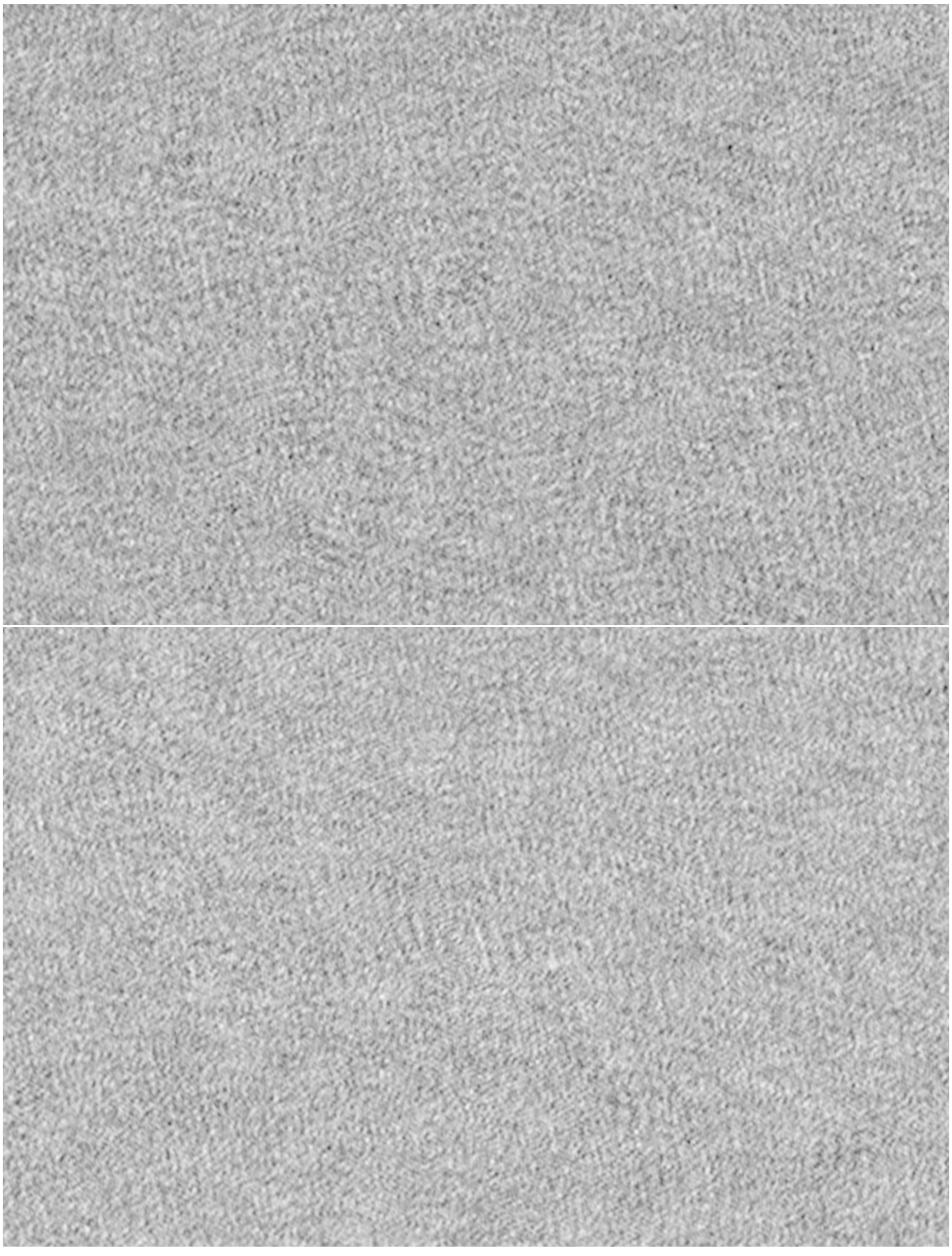
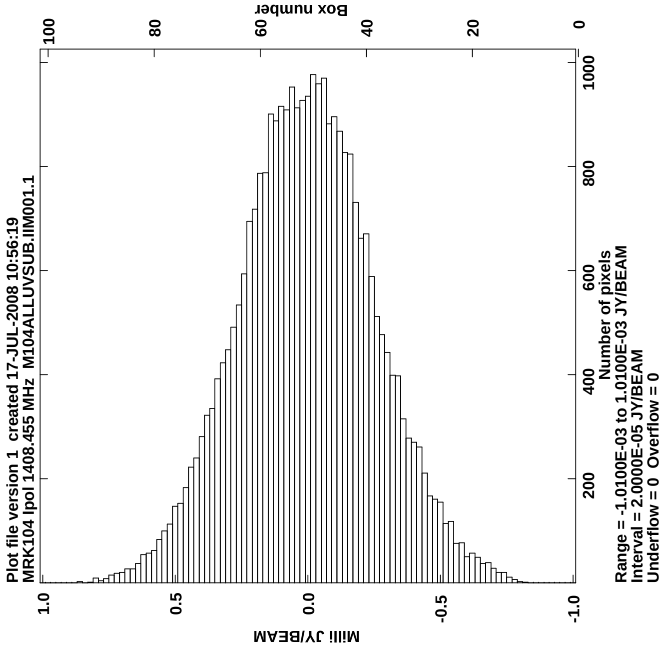


Figure 5: The images made after the bad baselines shown in Fig. 4 have been removed from the UV data. The top panel is Stokes RR and the bottom panel is Stokes LL. Notice that the intense concentric ring-like artifacts are not present in the image although a lower level ring-like structure does seem to be present in the Stokes RR image which is another bad baseline in the data.



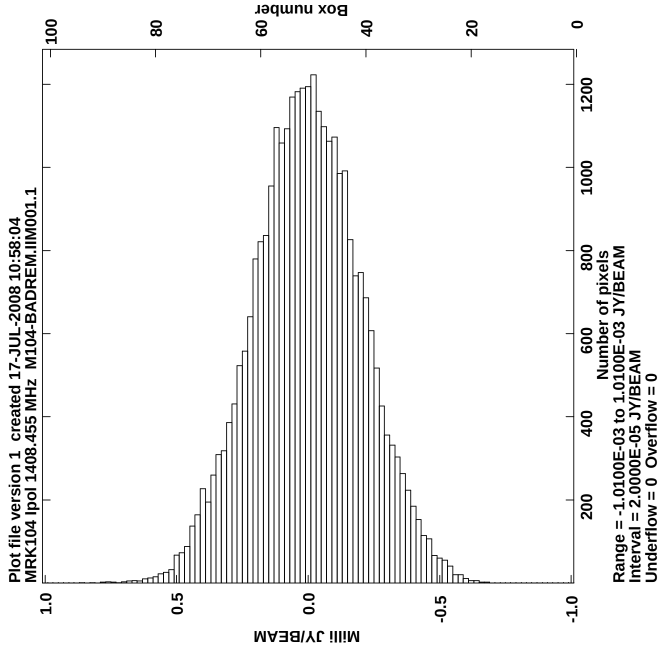


Figure 6: The histogram showing the noise statistics of the image before the bad baselines are removed (top panel) and after the bad baselines are removed (bottom panel). The lower rms ($\sim 200\mu$Jy) of the bottom panel compared to that of the upper panel rms ($\sim 270\mu$Jy) indicates that bad data has been removed.

6 Appendix A - Recommended recipe:

In this appendix, we suggest how to use the programme *badbase*:

1. Remove (UVSUB) all clean components from the uv data and make a residual image which will clearly show the ring-like structure. Image RR and LL separately. Make a large image say 2048 × 2048.

2. Take an FFT of the above image which will result in the complex UV data being shown in an image format. The real and imaginary uv images will be created. Examine the real part of the result.

3. . If there are bad baselines they will generally appear as bright/faint tracks compared to the rest. Moreover they will not show a noise-like behavioiur. You need to identify these to be able to flag them.

4. To find the U and V coordinates of any two points on the bad track in AIPS, use either:
   a. curval to find the x, y coordinates.
   b. load x,y into the adverb PIXXY and run the verb IMVAL c. Note down the UU and VV coordinates that you get (note that IMVAL shows VV first and then UU).
   or:
   a. define a small box using TVWIN around the track enclosing the point you want to obtain the coordinates for.
   b. run IMEAN and note down the UU and VV coordinates of the maximum or minimum point whichever is the relevant number on the track.
   c. repeat a, b for another point on the track.

   5. Run the programme 'badbase' that we have developed using the algorithm. If (4) has been done carefully, the programme will identify the track and print the bad baseline and how close it is to the baseline identified from the bad data. Better the identification in (5), better will be the match. The programme can be run as follows and takes the coordinates of the two points, declination of the source in degrees, wavelength in cms and the desired tolerance in metres.

   syntax:badbase U1 V1 U2 V2 dec_deg lamdba_cm tolerance_m

   For example, for identifying the bad short baseline in Fig.4 the inputs we gave to the programme were:

   badbase 22.057 14.402 13.697 25.481 53.44 21.26 50

The results we obtained were as follows:

OFFSET= 7.5m = 0.04 klambda

EQUA=4759.010m POL=4017.030m BadBaseline=13-27

EQUA_DATA=4763.500m POL_DATA=4011.055m

This meant that the bad baseline was 13-27 ie. C13-W03 and the difference between the baseline estimated from the coordinates that we obtained from the UV tracks (EQUA, POL) and those estimated from the antenna positions (EQUA_DATA, POL_DATA) was only 7.5m.

7 Appendix B: the programme in C

The *badbase* programme is listed below on the first two pages. This programme uses the file UVBASELINE.TXT which contains the equatorial and polar coordinates for all the 435 baselines as detailed in the section on the algorithm. The next seven pages give these coordinates. We estimate these values from the antenna positions as detailed in the algorithm. The antenna positions used to arrive at these values are given on the last page after UVBASELINE.TXT. The first column lists the antenna name whereas the second, third and fourth column give the values of B_x, B_y and B_z in metres.

```
/*  A short programme to identify bad baselines based on RN's alg
o - NGK 10/7/2008

    To compile:
    gcc -o badbase badbase.c -lm

*/

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <string.h>


main(int argc, char *argv[])
{

FILE *fp;
float X1_klambda, Y1_klambda, lambda_cm, Xm, Ym, X1m,X2m,Y1m,Y2m;
float X2_klambda, Y2_klambda, t, t1;
float *EQUA, *POL, EQ, PO, DIFF, FINAL, delta;
int   *n, *m, A1, A2, i, j;
char  tmp[40], tmp1[40], tmp2[40], tmp3[40], tmp4[40] ;

 if (argc != 8)
     {
        printf("\n Input the coordinates of any two points lying on the bad baseline (uv) t
rack. \n");
        printf(" FFT of the corrupted image will generate the uv image from which the a
bove can be identified.\n");
        printf("\n Syntax: badbase <UU1 klambda> <VV1 klambda> <UU2 klambda>
<VV2 klambda> <declination deg> <lambda cm> <tolerance m> \n\n");
        exit(1);
     }

   X1_klambda  = atof(argv[1]);
   Y1_klambda  = atof(argv[2]);
   X2_klambda  = atof(argv[3]);
   Y2_klambda  = atof(argv[4]);
   delta = atof(argv[5]);
   lambda_cm  = atof(argv[6]);
   FINAL = atof(argv[7]);

   delta = delta* 3.1415927/180.0;

EQUA      = (float *)calloc(500,sizeof(float));
POL       = (float *)calloc(500,sizeof(float));
n         = (int *)calloc(500,sizeof(int));
m         = (int *)calloc(500,sizeof(int));

  if ((fp=fopen("UVBASELINE.TXT","r"))==NULL)   {
      printf("\nUVBASELINE.TXT : Unable to open"); exit(1); }
```

```
   X1m = X1_klambda*(lambda_cm*10);    /* in metres */
   Y1m = Y1_klambda*(lambda_cm*10);
   X2m = X2_klambda*(lambda_cm*10);
   Y2m = Y2_klambda*(lambda_cm*10);

  fscanf(fp,"%s %s %s %s %s\n", tmp, tmp1, tmp2, tmp3, tmp4);

  t = (Y2m*Y2m - Y1m*Y1m);
  t1 = (X1m*X1m - X2m*X2m)*sin(delta)*sin(delta);
  Ym = (t - t1)/(2*(Y2m - Y1m));
  t = X1m*X1m;  t1 = (Y1m-Ym)*(Y1m-Ym);
  Xm = sqrt(t + t1/pow(sin(delta),2));
  Ym = Ym/cos(delta);

  for (i=0;i<435;i++)
    {  fscanf(fp, "%d %d  %f  %f", &n[i], &m[i],&EQUA[i], &POL[i]);

    }

  DIFF = 0.0;
  for (i=0;i<435;i++)
     {
         DIFF = sqrt(pow((EQUA[i] - abs(Xm)),2) + pow((POL[i] - a
bs(Ym)),2));
         if (DIFF<FINAL)
          {
               EQ = EQUA[i]; PO = POL[i];  A1 = n[i], A2 = m[i];
   printf("\n\n OFFSET=%5.1fm = %4.2f klambda\t \n", DIFF,DIFF/(lambda_cm*
10));
   printf("\n  EQUA=%5.3fm\t POL=%5.3fm\t BadBaseline=%d-%d\n", EQ,PO,A1,
A2);
   printf("\n  EQUA_DATA=%5.3fm \t POL_DATA=%5.3fm\n", Xm, Ym);
            }
    }
}
```

BASELINE	EQUA m	POL m
1 2	361.504730	20.309998
1 3	687.915100	20.040001
1 4	1062.187500	153.630005
1 5	1255.162231	143.470001
1 6	624.242432	226.549988
1 7	722.186401	200.540009
1 8	425.618805	380.289978
1 9	647.298767	131.609985
1 10	872.463135	567.450012
1 11	1294.684814	301.519989
1 12	551.486389	615.500000
1 13	1375.788330	1097.880005
1 14	1178.741455	608.590027
1 15	2156.094482	973.709961
1 16	3953.230469	1952.500000
1 17	7169.240723	2923.330078
1 18	9585.564453	3363.239990
1 19	11494.449219	4563.169922
1 20	937.593994	2785.889893
1 21	1789.980835	4259.120117
1 22	2206.312988	6384.919922
1 23	3076.897949	8959.459961
1 24	4705.982910	13362.440430
1 25	2289.329590	611.359985
1 26	3818.807861	1439.430054
1 27	5971.923828	2919.150146
1 28	7920.729492	5087.569824
1 29	9200.132812	7837.180176
1 30	12331.836914	8936.299805
2 3	326.698395	40.349998
2 4	702.104248	173.940002
2 5	894.684998	163.779999
2 6	266.862030	206.239990
2 7	362.537964	180.230011
2 8	126.124054	359.979980
2 9	286.693970	111.299995
2 10	522.587769	547.140015
2 11	933.977417	281.209991
2 12	247.806732	595.190002
2 13	1029.244995	1077.570068
2 14	823.336670	588.280029
2 15	2514.212646	994.019958
2 16	4310.270020	1972.809937
2 17	7527.896484	2943.640137
2 18	9945.095703	3383.550049
2 19	11853.418945	4583.479980
2 20	979.277893	2765.579834
2 21	1598.041992	4238.810059
2 22	2171.310059	6364.609863
2 23	3122.054932	8939.150391
2 24	4631.978516	13342.130859
2 25	1930.351440	631.669983
2 26	3461.546631	1459.739990
2 27	5617.014160	2939.460205
2 28	7569.991211	5107.879883
2 29	8857.327148	7857.490234
2 30	11984.042969	8956.609375
3 4	376.206604	133.589996
3 5	568.240479	123.430000
3 6	104.186272	246.589996
3 7	77.860046	220.580002
3 8	309.639221	400.329987

3	9	64.151070	151.649994
3	10	252.564377	587.489990
3	11	611.912292	321.559998
3	12	272.709808	635.539978
3	13	738.139465	1117.920044
3	14	517.082642	628.630005
3	15	2835.987305	953.669983
3	16	4630.382812	1932.459961
3	17	7849.505371	2903.290039
3	18	10267.726562	3343.199951
3	19	12175.281250	4543.129883
3	20	1136.259033	2805.929932
3	21	1498.560059	4279.160156
3	22	2209.779297	6404.959961
3	23	3215.675049	8979.500000
3	24	4607.512695	13382.480469
3	25	1604.641724	591.320007
3	26	3136.767822	1419.390015
3	27	5293.679688	2899.110107
3	28	7249.593262	5067.529785
3	29	8543.045898	7817.140137
3	30	11665.616211	8916.259766
4	5	193.220001	10.159996
4	6	459.374481	380.179993
4	7	362.541962	354.169983
4	8	678.229492	533.919983
4	9	426.448639	285.239990
4	10	319.319458	721.079956
4	11	276.995148	455.149994
4	12	606.325562	769.130005
4	13	497.311646	1251.510010
4	14	277.433838	762.219971
4	15	3201.072266	820.079956
4	16	4992.073242	1798.869995
4	17	8212.813477	2769.699951
4	18	10632.539062	3209.609863
4	19	12538.678711	4409.540039
4	20	1414.757568	2939.520020
4	21	1503.959717	4412.750000
4	22	2344.391113	6538.549805
4	23	3391.545898	9113.089844
4	24	4643.811523	13516.070312
4	25	1228.461426	457.730011
4	26	2760.632324	1285.800049
4	27	4918.025879	2765.520020
4	28	6875.586426	4933.939941
4	29	8173.628906	7683.550293
4	30	11292.836914	8782.669922
5	6	646.989929	370.019989
5	7	548.315613	344.010010
5	8	865.920288	523.760010
5	9	615.972351	275.079987
5	10	468.622314	710.919983
5	11	157.976349	444.989990
5	12	785.211548	758.969971
5	13	426.063385	1241.350098
5	14	274.347595	752.059998
5	15	3393.508301	830.239990
5	16	5183.680176	1809.029907
5	17	8404.670898	2779.860107
5	18	10824.693359	3219.770020
5	19	12730.497070	4419.699707
5	20	1558.152466	2929.359863

5	21	1516.991455	4402.590332
5	22	2409.594238	6528.390137
5	23	3471.317871	9102.929688
5	24	4647.962402	13505.910156
5	25	1036.967773	467.890015
5	26	2569.968750	1295.959961
5	27	4728.523438	2775.680176
5	28	6688.419922	4944.099609
5	29	7991.139648	7693.709961
5	30	11106.930664	8792.830078
6	7	99.570717	26.009995
6	8	219.034134	153.739990
6	9	40.036118	94.940002
6	10	258.350281	340.899994
6	11	671.505432	74.970001
6	12	168.537415	388.949982
6	13	764.296082	871.330078
6	14	556.500244	382.040009
6	15	2779.742188	1200.260010
6	16	4576.301758	2179.050049
6	17	7793.219238	3149.880127
6	18	10209.788086	3589.790039
6	19	12118.549805	4789.719727
6	20	1032.856445	2559.339844
6	21	1441.022461	4032.570068
6	22	2122.087646	6158.370117
6	23	3120.386719	8732.910156
6	24	4534.711914	13135.890625
6	25	1683.310425	837.910034
6	26	3216.662842	1665.979980
6	27	5375.511719	3145.700195
6	28	7334.616699	5314.119629
6	29	8632.592773	8063.729980
6	30	11752.169922	9162.849609
7	8	317.738373	179.749985
7	9	76.782928	68.930008
7	10	179.521149	366.909973
7	11	572.690002	100.979996
7	12	248.220398	414.959961
7	13	676.897705	897.340027
7	14	462.728638	408.049988
7	15	2876.707520	1174.250000
7	16	4672.798828	2153.040039
7	17	7890.360840	3123.870117
7	18	10307.345703	3563.780029
7	19	12215.833008	4763.709961
7	20	1096.662842	2585.349854
7	21	1421.856445	4058.580078
7	22	2144.508789	6184.379883
7	23	3157.286621	8758.919922
7	24	4534.065918	13161.900391
7	25	1584.153320	811.900024
7	26	3117.570312	1639.969971
7	27	5276.689941	3119.690186
7	28	7236.515625	5288.109863
7	29	8536.139648	8037.720215
7	30	11654.503906	9136.839844
8	9	252.507538	248.679993
8	10	449.645538	187.160004
8	11	884.404541	78.769989
8	12	131.763779	235.209991
8	13	950.432373	717.590088
8	14	758.251953	228.300018

8	15	2578.721924	1354.000000
8	16	4376.378906	2332.790039
8	17	7590.404785	3303.620117
8	18	10005.215820	3743.530029
8	19	11914.991211	4943.459961
8	20	885.712646	2405.599854
8	21	1472.331665	3878.830078
8	22	2054.437256	6004.629883
8	23	3016.752686	8579.169922
8	24	4508.994629	12982.150391
8	25	1901.859985	991.650024
8	26	3435.294678	1819.719971
8	27	5594.397461	3299.440186
8	28	7553.636719	5467.859863
8	29	8850.755859	8217.469727
8	30	11971.065430	9316.589844
9	10	251.289993	435.839996
9	11	647.444214	169.910004
9	12	208.569656	483.889984
9	13	752.852539	966.270020
9	14	539.511536	476.980011
9	15	2800.851562	1105.319946
9	16	4596.697754	2084.109863
9	17	7814.553223	3054.939941
9	18	10231.788086	3494.849854
9	19	12140.097656	4694.779785
9	20	1072.599121	2654.280029
9	21	1462.700073	4127.510254
9	22	2155.726318	6253.310059
9	23	3157.022705	8827.849609
9	24	4562.695312	13230.830078
9	25	1652.884888	742.969971
9	26	3185.931396	1571.040039
9	27	5344.105469	3050.760010
9	28	7302.008789	5219.179688
9	29	8598.251953	7968.790039
9	30	11718.994141	9067.910156
10	11	447.322906	265.929993
10	12	340.204407	48.049988
10	13	506.669281	530.430054
10	14	309.243134	41.140015
10	15	3027.856201	1541.159912
10	16	4825.346191	2519.949951
10	17	8039.970215	3490.780029
10	18	10454.857422	3930.689941
10	19	12364.621094	5130.620117
10	20	1096.854248	2218.439941
10	21	1277.666504	3691.670166
10	22	2054.468506	5817.469727
10	23	3088.861084	8392.009766
10	24	4406.122559	12794.990234
10	25	1480.137695	1178.810059
10	26	3010.934814	2006.880005
10	27	5172.204102	3486.600098
10	28	7138.831055	5655.019531
10	29	8450.491211	8404.629883
10	30	11559.802734	9503.750000
11	12	785.433289	313.979980
11	13	268.617218	796.360046
11	14	166.697968	307.070007
11	15	3447.386963	1275.229980
11	16	5242.236328	2254.020020
11	17	8461.056641	3224.850098

11	18	10878.820312	3664.760010
11	19	12786.756836	4864.689941
11	20	1495.734497	2484.369873
11	21	1370.893066	3957.600098
11	22	2282.958740	6083.399902
11	23	3350.968750	8657.940430
11	24	4496.395508	13060.920898
11	25	1034.318970	912.880005
11	26	2563.785156	1740.949951
11	27	4725.112793	3220.670166
11	28	6692.782227	5389.089844
11	29	8008.462891	8138.700195
11	30	11114.314453	9237.819336
12	13	829.814026	482.380066
12	14	648.563049	6.909973
12	15	2697.892090	1589.209961
12	16	4495.637207	2568.000000
12	17	7707.288086	3538.830078
12	18	10120.594727	3978.739990
12	19	12031.103516	5178.669922
12	20	865.491516	2170.389893
12	21	1356.610840	3643.620117
12	22	1981.583496	5769.419922
12	23	2966.033936	8343.959961
12	24	4417.043457	12746.940430
12	25	1813.914795	1226.859985
12	26	3346.576172	2054.929932
12	27	5507.361816	3534.650146
12	28	7471.180664	5703.069824
12	29	8776.350586	8452.679688
12	30	11890.835938	9551.799805
13	14	231.316925	489.290039
13	15	3527.635742	2071.590088
13	16	5325.398926	3050.379883
13	17	8536.748047	4021.209961
13	18	10949.217773	4461.120117
13	19	12860.144531	5661.049805
13	20	1397.004517	1688.009888
13	21	1117.922485	3161.240234
13	22	2059.996094	5287.040039
13	23	3135.660400	7861.580078
13	24	4232.781250	12264.560547
13	25	1109.989136	1709.239990
13	26	2603.005127	2537.310059
13	27	4759.010254	4017.030273
13	28	6736.227051	6185.449707
13	29	8072.107422	8935.060547
13	30	11159.498047	10034.179688
14	15	3334.824707	1582.300049
14	16	5131.917480	2561.089844
14	17	8347.674805	3531.919922
14	18	10763.150391	3971.830078
14	19	12672.606445	5171.759766
14	20	1329.693115	2177.299805
14	21	1250.087158	3650.530273
14	22	2135.487549	5776.330078
14	23	3198.716064	8350.870117
14	24	4386.658691	12753.850586
14	25	1190.611450	1219.949951
14	26	2714.840576	2048.020020
14	27	4875.983398	3527.740234
14	28	6846.474121	5696.159668
14	29	8166.755371	8445.770508

14	30	11268.797852	9544.889648
15	16	1797.769409	978.789978
15	17	5013.717285	1949.620117
15	18	7431.867676	2389.530029
15	19	9339.370117	3589.459961
15	20	2534.147949	3759.599854
15	21	3656.142090	5232.830078
15	22	3545.687988	7358.629883
15	23	3897.147949	9933.169922
15	24	5877.606445	14336.150391
15	25	4408.925781	362.349976
15	26	5915.492188	465.720032
15	27	8040.280273	1945.440186
15	28	9950.650391	4113.859863
15	29	11169.428711	6863.470215
15	30	14327.341797	7962.589844
16	17	3221.587646	970.830078
16	18	5643.587402	1410.739990
16	19	7547.049316	2610.669922
16	20	4273.684082	4738.389648
16	21	5394.781738	6211.620117
16	22	5134.895508	8337.419922
16	23	5239.888672	10911.959961
16	24	7248.895996	15314.940430
16	25	6188.666992	1341.139893
16	26	7678.704590	513.069946
16	27	9780.037109	966.650146
16	28	11660.342773	3135.069824
16	29	12835.667969	5884.680176
16	30	16001.774414	6983.799805
17	18	2423.372559	439.909912
17	19	4325.881836	1639.839844
17	20	7416.003906	5709.219727
17	21	8519.955078	7182.450195
17	22	8114.596191	9308.250000
17	23	7973.824707	11882.790039
17	24	9905.199219	16285.770508
17	25	9409.805664	2311.969971
17	26	10894.217773	1483.900024
17	27	12980.667969	4.179932
17	28	14836.122070	2164.239746
17	29	15971.262695	4913.850098
17	30	19138.023438	6012.969727
18	19	1914.431763	1199.929932
18	20	9798.255859	6149.129883
18	21	10889.714844	7622.360352
18	22	10423.988281	9748.160156
18	23	10181.840820	12322.700195
18	24	12041.529297	16725.679688
18	25	11832.250977	2751.879883
18	26	13317.541992	1923.809937
18	27	15401.015625	444.089844
18	28	17248.031250	1724.329834
18	29	18366.859375	4473.940430
18	30	21531.798828	5573.059570
19	20	11712.592773	7349.059570
19	21	12803.050781	8822.290039
19	22	12326.564453	10948.089844
19	23	12056.176758	13522.629883
19	24	13885.642578	17925.609375
19	25	13733.883789	3951.809814
19	26	15211.875000	3123.739746
19	27	17283.072266	1644.019775

19	28	19113.187500	524.399902
19	29	20208.474609	3274.010254
19	30	23369.250000	4373.129883
20	21	1123.565186	1473.230225
20	22	1277.490234	3599.030029
20	23	2152.856689	6173.570312
20	24	3785.020996	10576.550781
20	25	2502.860352	3397.250000
20	26	3996.280518	4225.319824
20	27	6146.354980	5705.040039
20	28	8126.707520	7873.459961
20	29	9468.464844	10623.070312
20	30	12549.335938	11722.189453
21	22	1012.833862	2125.799805
21	23	2086.529541	4700.339844
21	24	3139.860352	9103.320312
21	25	2058.304443	4870.479980
21	26	3348.232666	5698.550293
21	27	5415.843750	7178.270508
21	28	7394.017578	9346.689453
21	29	8782.929688	12096.300781
21	30	11793.432617	13195.419922
22	23	1080.234741	2574.540039
22	24	2508.422852	6977.520508
22	25	3065.729248	6996.279785
22	26	4346.913086	7824.350098
22	27	6379.951660	9304.070312
22	28	8349.043945	11472.490234
22	29	9751.258789	14222.099609
22	30	12728.446289	15321.219727
23	24	2010.894165	4402.980469
23	25	4143.803223	9570.820312
23	26	5387.089355	10398.889648
23	27	7369.932617	11878.610352
23	28	9321.948242	14047.029297
23	29	10738.409180	16796.640625
23	30	13667.710938	17895.759766
24	25	4947.720215	13973.800781
24	26	5763.193848	14801.870117
24	27	7384.192871	16281.590820
24	28	9193.637695	18450.009766
24	29	10634.313477	21199.621094
24	30	13323.280273	22298.740234
25	26	1533.467407	828.070007
25	27	3693.545898	2307.790039
25	28	5658.712891	4476.209961
25	29	6976.286133	7225.820312
25	30	10080.013672	8324.939453
26	27	2161.362061	1479.720093
26	28	4133.740234	3648.139648
26	29	5475.473633	6397.750000
26	30	8556.978516	7496.869629
27	28	1983.371948	2168.419678
27	29	3371.470703	4918.030273
27	30	6403.358887	6017.149414
28	29	1440.902222	2749.610352
28	30	4423.285156	3848.729980
29	30	3167.324951	1099.119629

C00:01	6.9500	687.8800	-20.0400
C01:02	13.2400	326.4300	-40.3500
C02:03	0.0000	0.0000	0.0000
C03:04	-51.1000	-372.7200	133.5900
C04:05	-51.0800	-565.9400	123.4300
C05:06	79.0900	67.8200	-246.5900
C06:07	71.2300	-31.4400	-220.5800
C08:08	130.7700	280.6700	-400.3300
C09:09	48.5600	41.9200	-151.6500
C10:10	191.3200	-164.8800	-587.4900
C11:11	102.4200	-603.2800	-321.5600
C12:12	209.2800	174.8500	-635.5400
C13:13	368.5800	-639.5300	-1117.9200
C14:14	207.3000	-473.7100	-628.6300
E02:15	-348.0400	2814.5500	953.6700
E03:16	-707.5800	4576.0000	1932.4600
E04:17	-1037.1100	7780.6900	2903.2900
E05:18	-1177.3700	10200.0000	3343.2000
E06:19	-1571.3200	12073.4600	4543.1300
S01:20	942.9900	633.9200	-2805.9300
S02:21	1452.8500	-367.3000	-4279.1600
S03:22	2184.5400	333.0300	-6404.9600
S04:23	3072.8600	947.6800	-8979.5000
S06:24	4592.7100	-369.0400	-13382.4800
W01:25	-201.5000	-1591.9400	591.3200
W02:26	-482.6700	-3099.4100	1419.3900
W03:27	-992.0100	-5199.9000	2899.1100
W04:28	-1734.5500	-7039.0300	5067.5300
W05:29	-2706.0900	-8103.1300	7817.1400
W06:30	-3102.1100	-11245.6000	8916.2600

8 Appendix C: Some hints for getting the most out of this programme:

Here are some useful hints if it gets difficult to use this programme to identify bad baselines. Although there are no gurantees, with experience we find that using different approaches as the data demands helps get rid of bad baselines.

- Examine the source-subtracted image for the presence of ring-like artifacts centred about the pointing centre. Only if you can notice them then you proceed with the procedure outlined above.
- Sometimes the bad baseline is lost in the maze of baselines in the uv image. Load only a small range close the maximum value in the uv image and check for any baseline which shows a constant offset instead of noise-like behvaiour. If none are visible then load only a small range of values close to the minimum in the uv image and check for any baselines which show a constant offset. This helps in the regions of the uv plane which are crowded. If you do identify the bad baseline then note down the U and V coordinates and use *badbase*.
- Inspite of doing the above at times it does not help identify the bad baseline. In this case, note down the rough U and V of the bad baseline from the uv image and then examine the data using other tasks such as VPLOT etc. This can also be done by noting down the fringe frequency of the ring-like artifact and finding out the uv baseline.
- Sometimes severe RFI can give rise to stripes in the image which confuse the ring-like structure - use the UVSUBed data to CLIP the bad data or remove bad data by identifying the culprits and then continue with the above procedure till the ring-like structure disappears.
- Lastly if all your attempts to flag that bad baseline fails...you can use the data if the bad baseline is at a very low level else request the Centre Director for makeup time on the telescope :>.