NATIONAL RADIO ASTRONOMY OBSERVATORY
Tucson, Arizona

INTEROFFICE MEMORANDUM

TO: 12 m Memo Series

FROM: T.K. White and P. R. Jewell

DATE: August 6, 1991

SUBJECT: Fourier Transforms of 12 m Structural Tilt Measurements

This is a follow-up to *12 m Memo* No. 247 (July 9, 1991) by P. R. Jewell and T. K. White on 12 m Structural Tilt Measurements (STM).

The discrete Fourier transforms (DFT) of certain data sets from the memo of July 9 were computed by the program `fourier.c`. The theory behind `fourier.c` and a listing of the program are given in the Appendix of this report. The DFT program computes the amplitudes and phases of the DFT of a data function. From the amplitude of the frequency points in the Fourier domain, one can see what trigonometric (cosine) terms may be present in the tilt curves.

This report contains the plots of the DFT amplitudes and phases for several "data functions" from the STM report. The equations in the Figure Captions below give the cosine amplitudes and phases of each DFT of the tilt data. In each equation, "A" represents the azimuth angle in degrees, the amplitude coefficient is in units of arcseconds, and the phase offset is in degrees. The amplitude at zero frequency corresponds to twice the absolute value of the "DC" offset; the phase at zero frequency is not plotted because it has no physical significance.

Figure 1 -- Tilts Above Az Bearing, 20 June 91 (corresponds to Fig. 1 in STM report):

$$13.8 \cdot \cos(A - 119.1) + 1.1 \cdot \cos(2 \cdot A - 170.4) + 0.7 \cdot \cos(4 \cdot A + 28.2)$$

Figure 2 -- DFT of Pedestal Tilts, below the azimuth bearing at the south position, 26 June 91 (from Fig. 3 of STM report):

$$0.92 \cdot \cos(A + 112) + 6.36 \cdot \cos(2 \cdot A - 157.7) + 0.66 \cdot \cos(4 \cdot A + 24.9)$$

Figure 3 -- DFT of Pedestal Tilts (*S pos.*), 6 July 78 (from Fig. 4 of STM):

$$1.36 \cdot \cos(A + 178.9) + 4.95 \cdot \cos(2 \cdot A + 19.6) + 1.33 \cdot \cos(4 \cdot A - 170.3)$$

Figure 4 -- DFT of Pedestal Tilts, 16 May 89 (from Fig. 5 of STM, *west position*):

0.95•cos(A-99.7) + 4.15•cos(2•A+5.82)

Figure 5 -- DFT of Pedestal Tilts, 16 May 89 (*north position*):

1.2•cos(A+154.8) + 4.6•cos(2•A-157.8) + 0.6•cos(4•A+45)

Figure 6 -- DFT of Pedestal Tilts, 16 May 89 (*east position*):

0.92•cos(A+82) + 3.75•cos(2•A+5.28)

Figure 7 -- DFT of Pedestal Tilts, 16 May 89 (*south position*):

0.99•cos(A+175.18) + 5.81•cos(2•A-159.47) + 1.05•cos(4•A+11.36)

Figure 8 -- DFT of Tilts of Inductosyn Box, 20 June 91 (from Fig. 6 of STM)

2.09•cos(A+1.44) + 2.94•cos(2•A-148.22)

Figure 9 -- DFT of Pedestal Floor Tilts, 20 June 91 (from Fig. 8 of STM)

2.45•cos(A+8.44)

Figure 10 -- DFT of East Yoke Arm Tilts, 15 May 89 (from Fig. 9 of STM) (*perp. pos.*):

14.5•cos(A-132.12) + 0.84•cos(2•A-179.01)

Figure 11 -- DFT of East Yoke Arm Tilts, 15 May 89, (‖ *pos.*):

13.95•cos(A+139.38) + 1.43•cos(2•A+113.94) + 0.89•cos(4•A-29.66)

APPENDIX

`fourier.c` and Discrete Fourier Transform Theory

`fourier.c` computes the discrete Fourier transform (DFT) of a set of up to 256 data points. We chose to use a DFT rather than a fast Fourier transform (FFT) because in this application we are usually transforming a small number of data points that are not usually an integral power of 2. The DFT at each frequency is computed by:

$$F(\sigma) = \sum_{t=0}^{N-1} f(t) \left[\cos\left[2\pi\sigma \frac{t}{N}\right] + i \sin\left[2\pi\sigma \frac{t}{N}\right] \right], \quad (1)$$

where f(t) is the "data function" with N data points.

This relation gives us the sinusoids that make up the data function. This is because to compute the transform at a given frequency (σ), all the data points are multiplied by a cosine and a sine of that same frequency. If the data function does not contain a sinusoid of the given frequency, then the transform tends to sum to zero at that frequency. We adopt the convention that the sinusoids are expressed as

$$\cos(N \cdot t - \Theta), \quad (2)$$

where N is an integer and positive Θ represents a positive phase offset.

In `fourier.c` the real components of the transform are stored in the even-numbered elements of an array, and the imaginary components are stored in the odd-numbered elements.

The subroutine `amp` computes the amplitudes of the transform

$$a(\sigma) = \frac{2}{n}\sqrt{R_\sigma^2 + I_\sigma^2}, \quad (3)$$

where R_σ is the real component of $F(\sigma)$, and I_σ is the imaginary component.

For our purposes, the second half of the amplitudes is just a mirror of the first half, excluding frequency point number zero. Here's why:

$$F(N-\sigma) = \sum_{t=0}^{N-1} f(t)\left[\cos\left(2\pi\frac{(N-\sigma)}{N}t\right) + i\sin\left(2\pi\frac{(N-\sigma)}{N}t\right)\right] - \sum_{t=0}^{N-1} f(t)\left[\cos\left(2\pi\frac{(-\sigma)}{N}t\right) + i\sin\left(2\pi\frac{(-\sigma)}{N}t\right)\right] = F(-\sigma) \quad (4)$$

but the amplitude of $(-\sigma)$ =

$$a(-\sigma) = \frac{2}{N}\sqrt{R^2_{(-\sigma)} + I^2_{(-\sigma)}} = \frac{2}{N}\sqrt{R^2_{\sigma} + I^2_{\sigma}} = a(\sigma) \quad (5)$$

The definition for the phase of the transform is

$$\Phi(\sigma) = \tan^{-1}\left[\frac{I_\sigma}{R_\sigma}\right] \quad (6)$$

For example, consider the case of a pure "1-Θ" cosinusoid of amplitude a and phase 90°, such that

$$a\cdot\cos(\Theta - 90) = a\cdot\sin(\Theta). \quad (7)$$

In this case, the first frequency point in the Fourier domain is given by

$$F(1) = \sum_{t=0}^{N-1} a\sin(t)\left[\cos\left[2\pi\frac{t}{N}\right] + i\sin\left[2\pi\frac{t}{N}\right]\right] \quad (8)$$

so that

$$R_1 = \sum_{t=0}^{N-1} a\sin(t)\cos\left[2\pi\frac{t}{N}\right] = 0, \quad (9)$$

and

$$I_1 = \sum_{t=0}^{N-1} a\sin^2(t) = \frac{a}{2}N, \quad (10)$$

where N data points complete one period of sin(Θ). Thus, as expected, the amplitude is given

by

$$\frac{2}{N}\sqrt{R_1^2 + I_1^2} - \frac{2}{N}\sqrt{0 + \left(\frac{a}{2}N\right)^2} - a \qquad (11)$$

and the phase is given by

$$\tan^{-1}\left[\frac{I_1}{R_1}\right] - \tan^{-1}\left[\frac{aN/2}{0}\right] \equiv 90°. \qquad (12)$$

To be considered significant, the amplitude (given by the subroutine amp) must be above the noise level. Phases for frequency points with negligible amplitudes may be ignored. To illustrate further, assume frequency points one and two have significant amplitudes; that is, our data set contains significant sin(Θ) and sin (2•Θ) functions. If phaz gives 135° for the phase of frequency point number one and 170° for the phase of frequency point number two, then the data set contains a cosinusoid of the form:

$$\begin{aligned} &a \cdot \cos(\Theta - 135°) + a \cdot \cos(2 \cdot \Theta - 170°) \\ &\equiv a \cdot \sin(\Theta - 45°) + a \cdot \sin(2 \cdot \Theta - 80°) \end{aligned} \qquad (13)$$

As long as the sinusoids in the data set trace an integer number of cycles and the data points are evenly spaced in the "time" domain, there is no need for windowing the data. However, if needed, windowing can be turned on during run time.

`fourier.c` assumes an input file consisting of two columns of data. The first column presumably contains the abscissas for the ordinates in the second column. In any case, the first column (the first field in each row) is ignored, and the second column is taken to be the data set to be Fourier transformed. The input file is read one line at a time, and the second field of each line is formatted as a double precision number.

A listing of `fourier.c` follows. The authors thank D. T. Emerson for help with Fourier transform theory.

KS/nc

```
/* Fourier.c computes the discrete Fourier transform (DFT) of a set of up to 256
   data points.  The number of data points need not be an integral power of 2.
   The subroutine "amp" computes the amplitudes of the transform.  For
   our purposes the second half of the amplitudes is just a mirror of
   the first half, excluding frequency point number zero.  The
   subroutine "phaz" computes the phases of the transform.
   For instance, if "phaz" gives -90 for frequency point number 2,
   then the original data set contains a cos(2*theta+90) curve.  This
   assumes that the amplitude at frequency point 2 is significant, i.e.,
   above the noise level.  Phases for frequency points with negligible
   amplitudes are meaningless.
        As long as the sinusoids in the data set trace an integer number
   of cycles and the data points are evenly spaced in the "time" domain,
   there is no need for windowing the data.  However, if needed, windowing
   can be turned on during run time.
        Fourier.c assumes an input file consisting of two columns of data.
   The first column presumably contains the abscissas for the ordinates in the
   second column.  In any case, the first column (the first field in each row)
   is ignored, and the second column is taken to be the data set to be Fourier
   transformed.  The input is formatted.  The input file is read one line at
   a time, and the first two fields of each line are formatted as doubles. */



#include <stdio.h>
#include <math.h>
#include <ctype.h>

#define MAXARRAY 512
#define MAXLINLEN 256

#define SWAP(a,b) tempr=(a); (a)=(b); (b)=tempr

main()
{
   double data[MAXARRAY];
   double tempr;/* temporary storage for swap */
   double win, arg;  /* used in window function */
   char *windat; /*window data--yes or no*/
   double ft[MAXARRAY], sum[MAXARRAY]; /*the transform(s) is/are stored in
                                         one or both of these*/
   int npts=0; /*number of ponts in the data set*/
   int c; /*loop counter*/
   double num;  /*dummy variable for first column of input file*/
   FILE *datfil; /*pointer to input file*/
   char line[MAXLINLEN],datfilename[MAXLINLEN];  /*input buffer*/
   char *cp; /*for lexical analysis of input */

   printf("\nEnter name of input file:\n");
   scanf("%s",datfilename);
   if ((datfil = fopen(datfilename,"r")) == NULL)
   {
      printf("Can't read data file %s--exiting.\n",datfilename);
      exit(1);
   }

   for (c=0; c<MAXARRAY; c++)   /* initializing */
   {
       data[c] = 0.0;
       sum[c] = 0.0;
       ft[c] = 0.0;
   }

   while (fgets(line,MAXLINLEN,datfil))
   { /* primitive lexical analysis--mostly to bloww of header lines*/
       for(cp = &line[0]; *cp; cp++)
```

```
        {
            if (isspace(*cp))
               continue;
            if (isalpha(*cp))
               break;
            if (isdigit(*cp) || (*cp == '-'))
            {
                 sscanf(line,"%lf  %lf",&num,&data[npts]);
                 npts++;
            }
            break;
         }
    }
    printf("\nWindow the data (y/n) ?\n");
    scanf("%s",line);
    windat = &line[0];
    if ((toupper(*windat)) == 'Y')
    {
       for (c=0; c<npts; c++)        /*window the data*/
       {
          arg = (c-0.5*(npts-1)) / (0.5*(npts+1));
          win = 1.0 - arg*arg;
          data[c] = data[c]*win;
       }
       four(data,sum,npts);              /* Fourier transform the data */
       for (c=0; c<npts/2; c++)        /* shift the data to counter the */
          SWAP(data[c],data[npts/2+c]); /* effects of the windowing the data */

       for (c=0; c<npts; c++)        /*window the shifted data*/
       {
          arg = (c-0.5*(npts-1)) / (0.5*(npts+1));
          win = 1.0 - arg*arg;
          data[c] = data[c]*win;
       }
       four(data,ft,npts);              /* Fourier transform the shifted data */
       for (c=0; c<2*npts; c++)         /* avg. the results with previous DFT */
          ft[c] = (sum[c] + ft[c])/2;
    }
    else /*don't window the data*/
       four(data,ft,npts);

    amp(ft,npts);
    phaz(ft,npts);
}


amp(avgft,npts)
double avgft[];  /* the result of a fourier transform */
int npts;
{
    double a[MAXARRAY];  /* the amplitude at each frequency point */
    int k;
    FILE *ampfil; /*points to the amplitudes output file*/
    char ampfilname[MAXLINLEN];

    for (k=0; k<MAXARRAY; k++)
        a[k] = 0.0;

    printf("\n/enter name of output file for amplitudes:\n");
    scanf("%s",ampfilname);
    if ((ampfil = fopen(ampfilname,"w")) == NULL)
    {
       printf("Can't open amplitude file %s--going on to phases.\n",ampfilname);
    }
    else
       for (k=0; k<npts; k++)
```

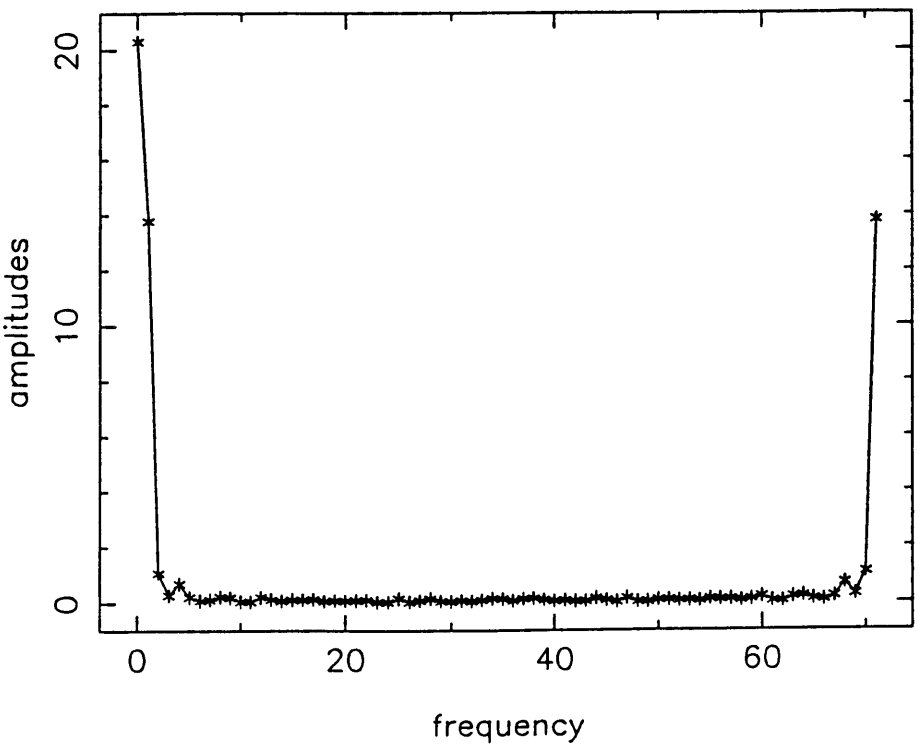
```
        {
            a[k] = ( sqrt( (avgft[2*k] * avgft[2*k])
                  + (avgft[2*k+1] * avgft[2*k+1]) ) ) / (npts/2);
            fprintf(ampfil, "%d  %5.8f\n",k,a[k]);
        }
}

phaz(avgft,npts)
double avgft[];
int npts;
{
   int h;
   double ph[MAXARRAY];  /* phase at each frequency point */
   FILE *phasfil;
   char phasfilname[MAXLINLEN];

   for (h=0; h<MAXARRAY; h++)
       ph[h] = 0.0;

   printf("\n/enter name of output file for phases:\n");
   scanf("%s",phasfilname);
   if ((phasfil = fopen(phasfilname,"w")) == NULL)
   {
      printf("Can't open phases file %s--exiting.\n",phasfilname);
      exit(1);
   }
   for (h=1; h<npts; h++)
   {
       ph[h] =180/3.1415927* atan2(avgft[2*h+1],avgft[2*h]);
       fprintf(phasfil, "%d %5.8f\n",h,ph[h]);
   }
}

four(data,ft,npts)
/* four() returns  (in the array ft[]) the discrete fourier transform (DFT) of
   the data set in the array data[]. The data is assumed to be real.  The
   transform is complex (except for special types of data functions) and is
   stored in the array ft[].  Real components are stored in even-numbered
   array elements and imaginary components are stored in odd-numbered
   elements.  */
double data[];
double ft[];
int npts; /* number  of points in the data array */
{
   int i,j;
   for (i=0; i<MAXARRAY; i++)   /* initializing */
       ft[i] = 0.0;
   for (i=0; i<(2*npts); i += 2)
   {
       for (j=0; j<npts; j++)
       {
           /* real component */
           ft[i] += data[j]*cos(3.14159*(double)(j*i)/(double) npts);
           /* imaginary */
           ft[i+1] += data[j]*sin(3.14159*(double) (j*i)/(double) npts);
       }
   }
}
```


DFT of Tilts Above Az Bearing, 20 June '91
amplitudes
0
10
20
0
20
40
60
frequency

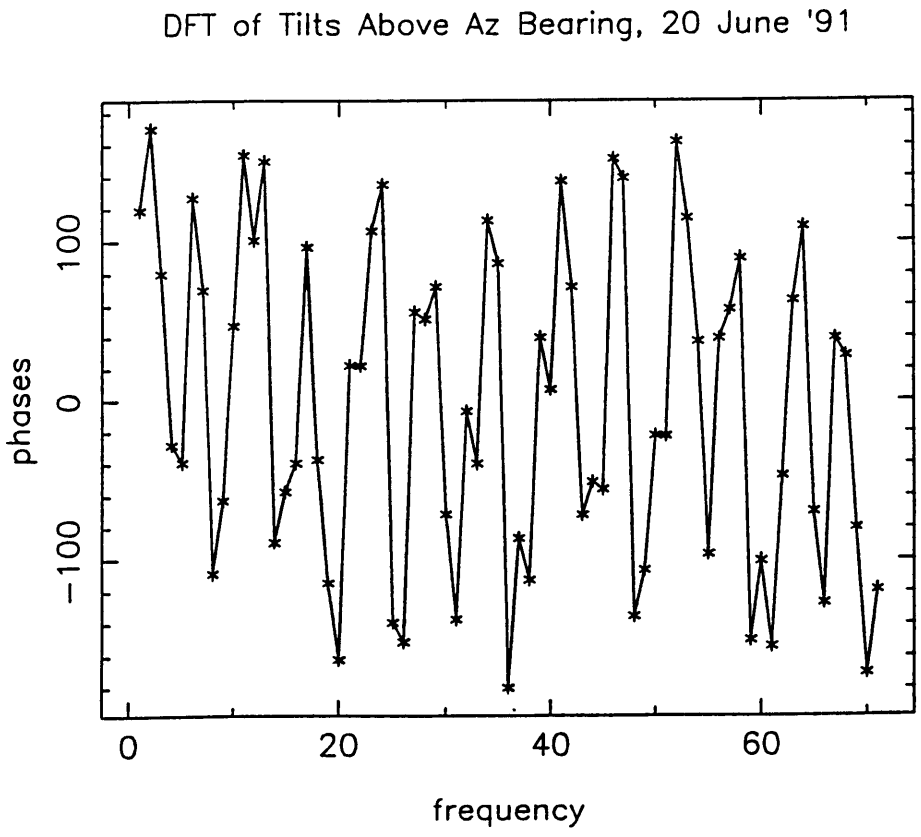
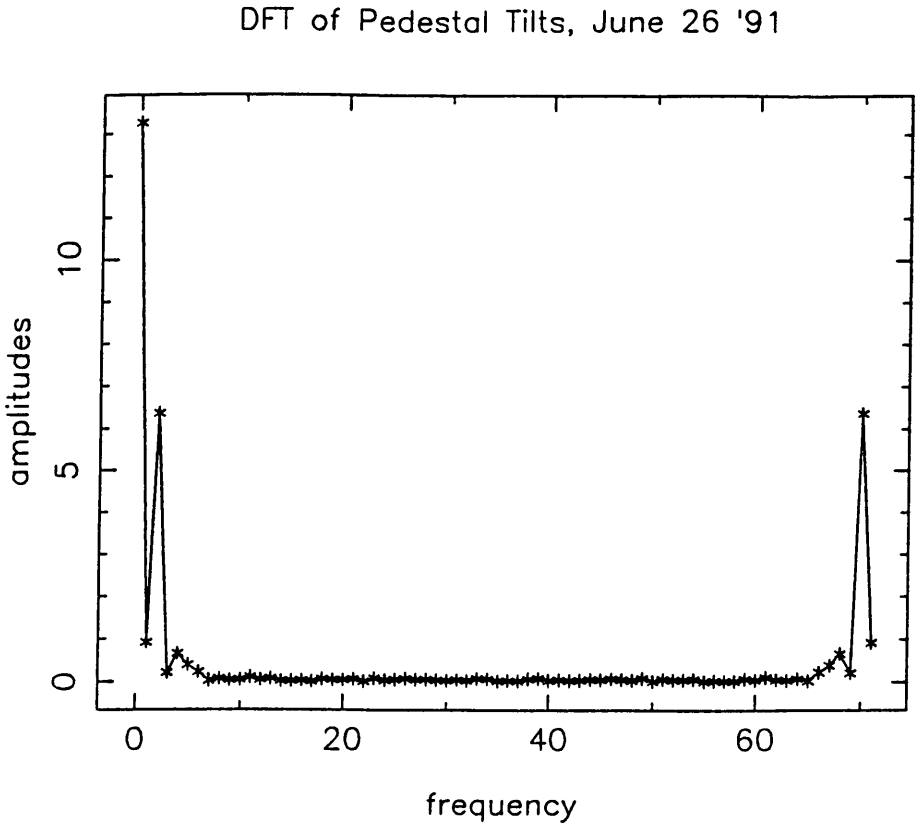

DFT of Tilts Above Az Bearing, 20 June '91
phases
-100
0
100
0
20
40
60
frequency

FIG. 1


DFT of Pedestal Tilts, June 26 '91
amplitudes
0
5
10
0
20
40
60
frequency

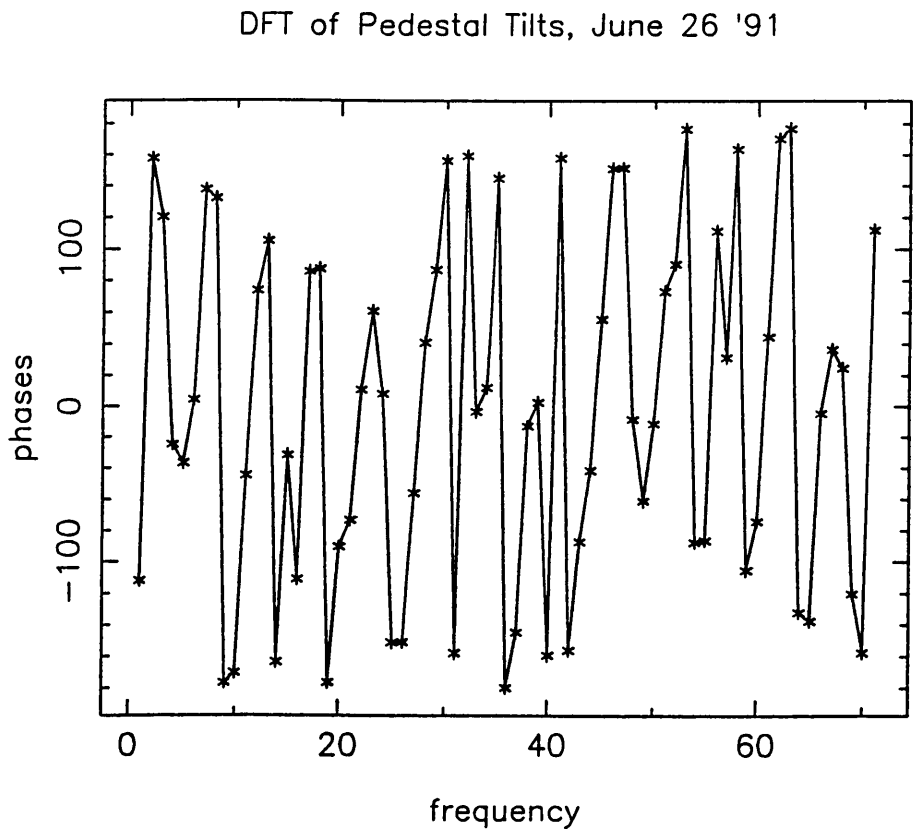
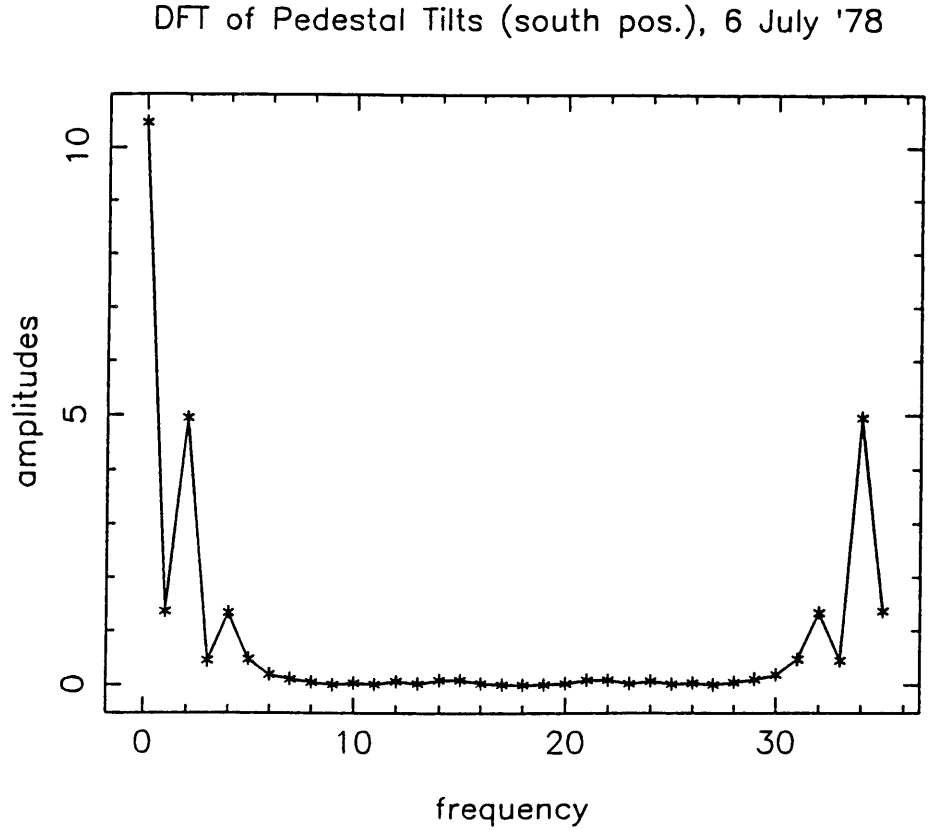

DFT of Pedestal Tilts, June 26 '91
phases
−100
0
100
0
20
40
60
frequency

FIG. 2


DFT of Pedestal Tilts (south pos.), 6 July '78
amplitudes
0
5
10
0
10
20
30
frequency

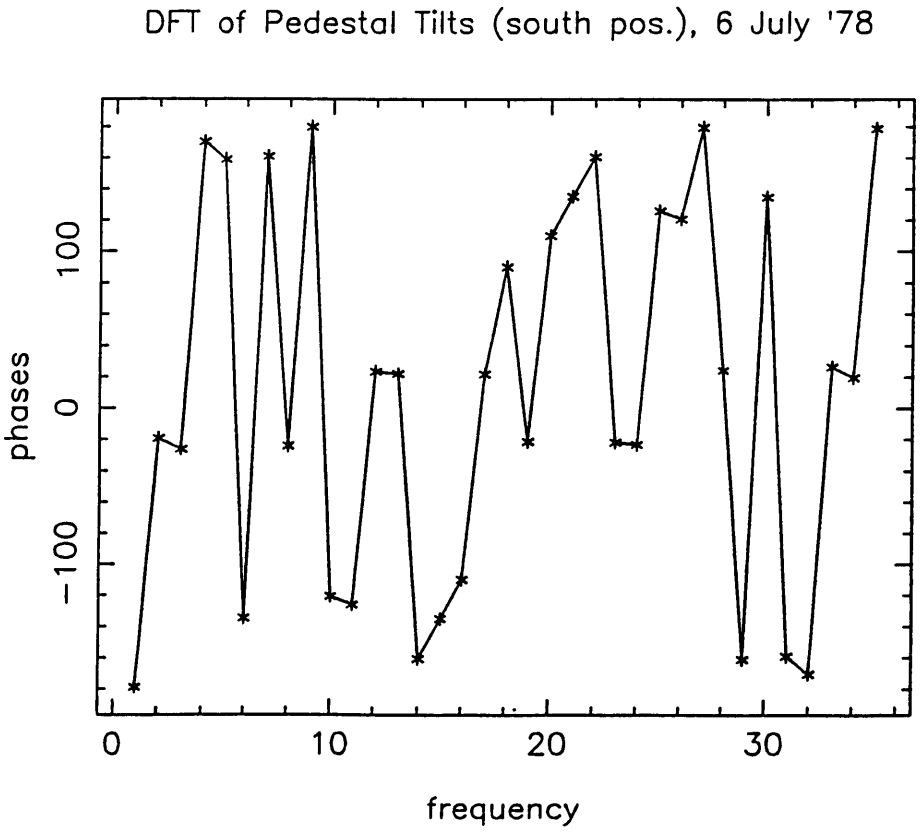
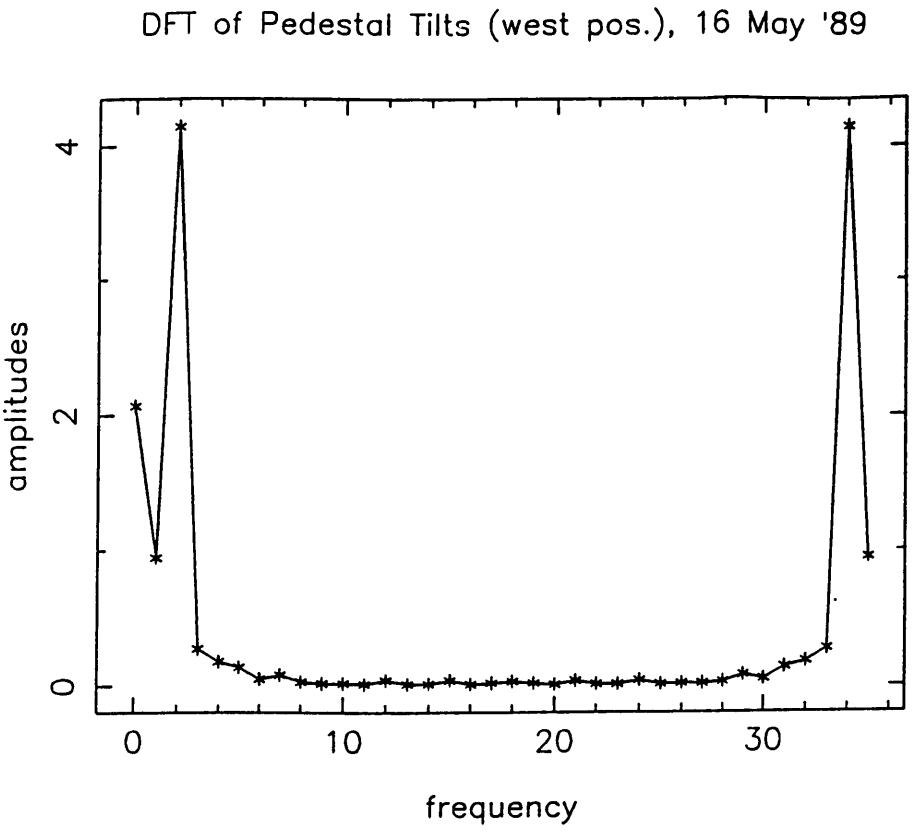

DFT of Pedestal Tilts (south pos.), 6 July '78
phases
-100
0
100
0
10
20
30
frequency

FIG. 3


DFT of Pedestal Tilts (west pos.), 16 May '89
amplitudes
0
2
4
0
10
20
30
frequency

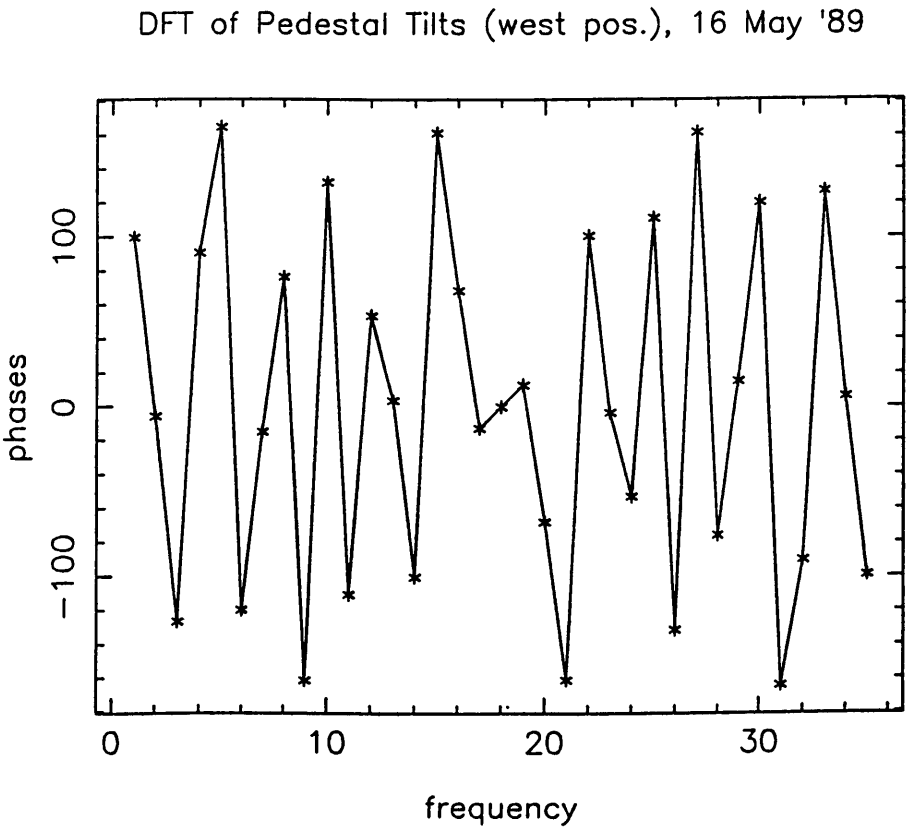
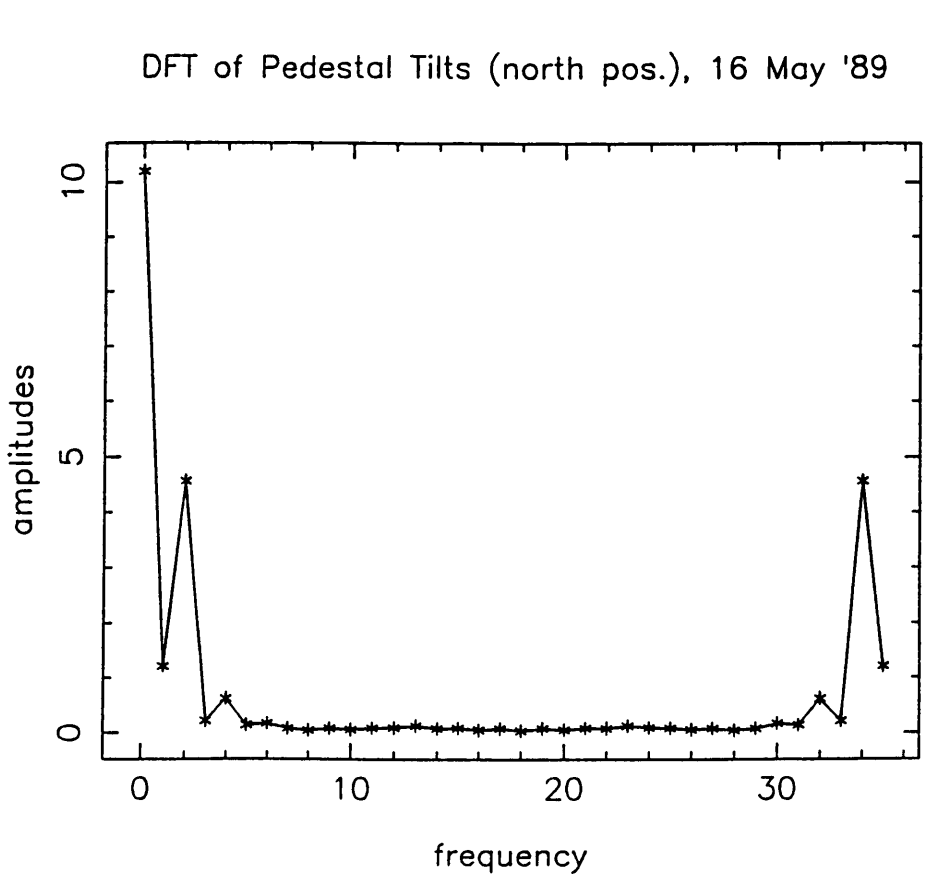

DFT of Pedestal Tilts (west pos.), 16 May '89
phases
-100
0
100
0
10
20
30
frequency

FIG. 4


DFT of Pedestal Tilts (north pos.), 16 May '89
amplitudes
0
5
10
0
10
20
30
frequency

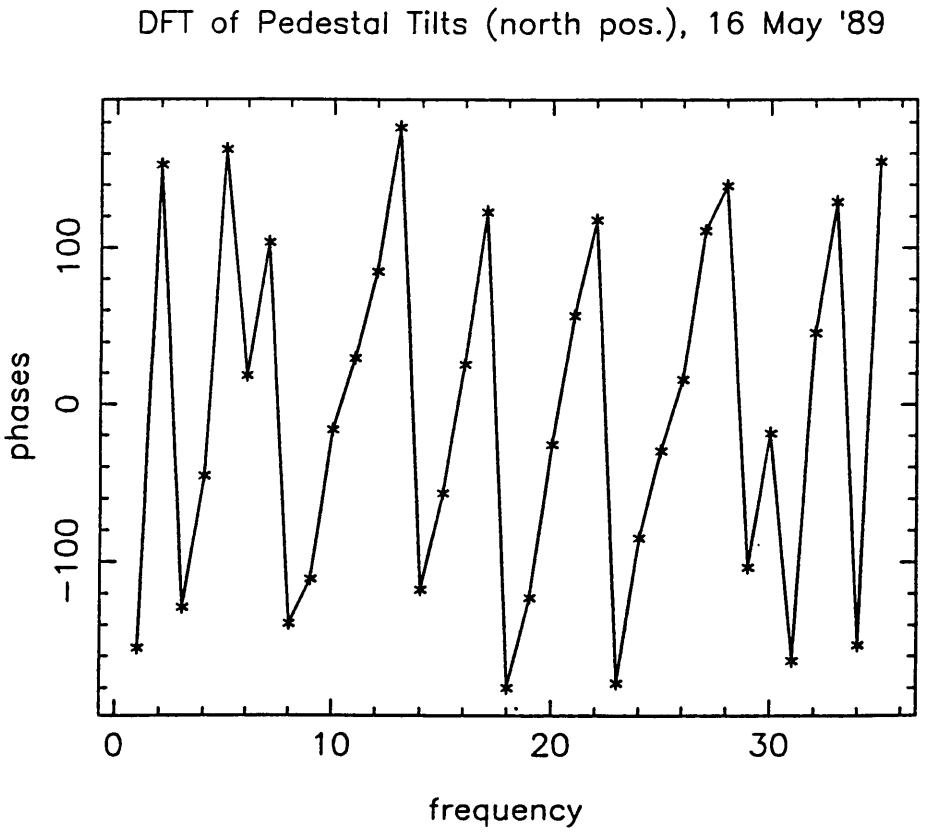
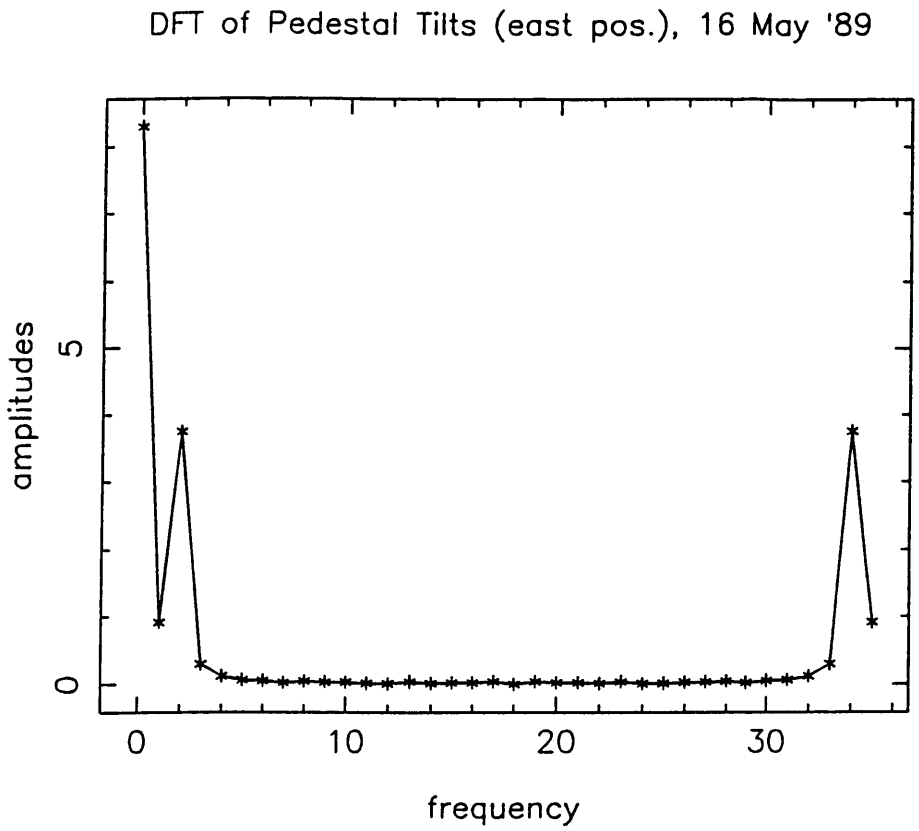

DFT of Pedestal Tilts (north pos.), 16 May '89
phases
-100
0
100
0
10
20
30
frequency

FIG. 5


DFT of Pedestal Tilts (east pos.), 16 May '89
amplitudes
0
5
0
10
20
30
frequency

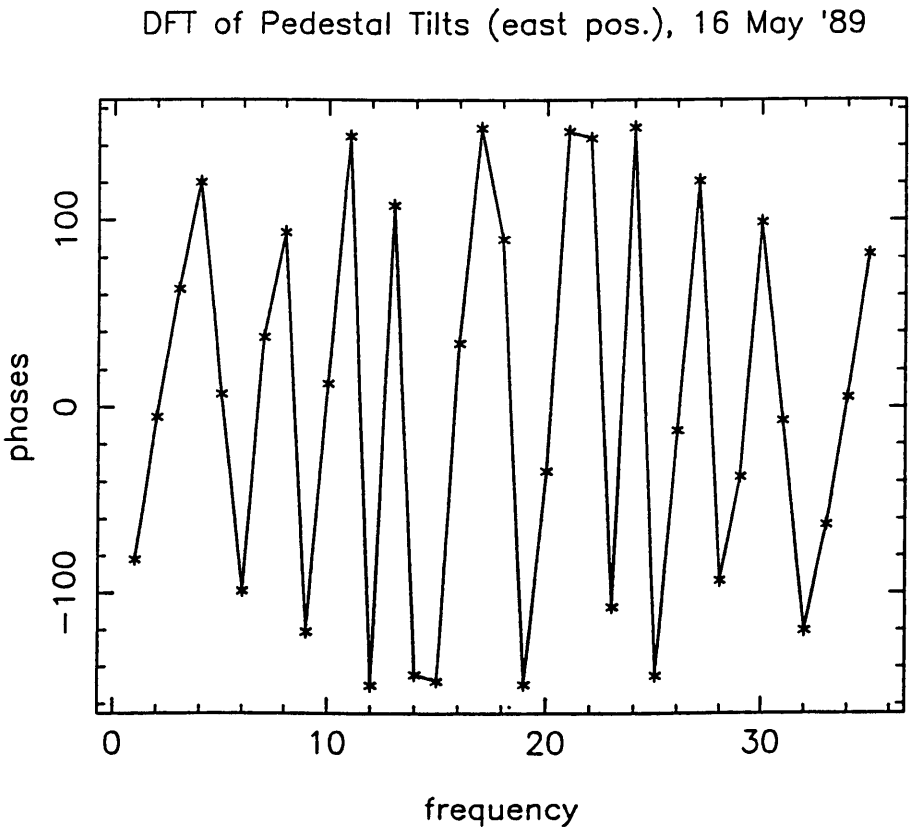
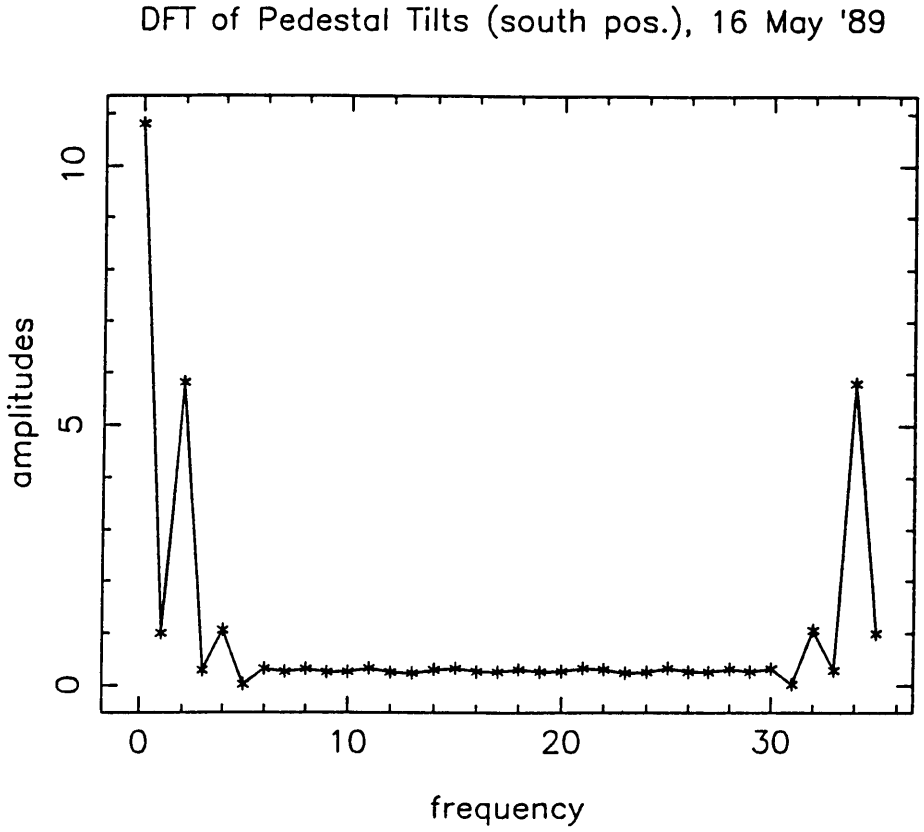

DFT of Pedestal Tilts (east pos.), 16 May '89
phases
-100
0
100
0
10
20
30
frequency

FIG. 6


DFT of Pedestal Tilts (south pos.), 16 May '89
amplitudes
0
5
10
0
10
20
30
frequency

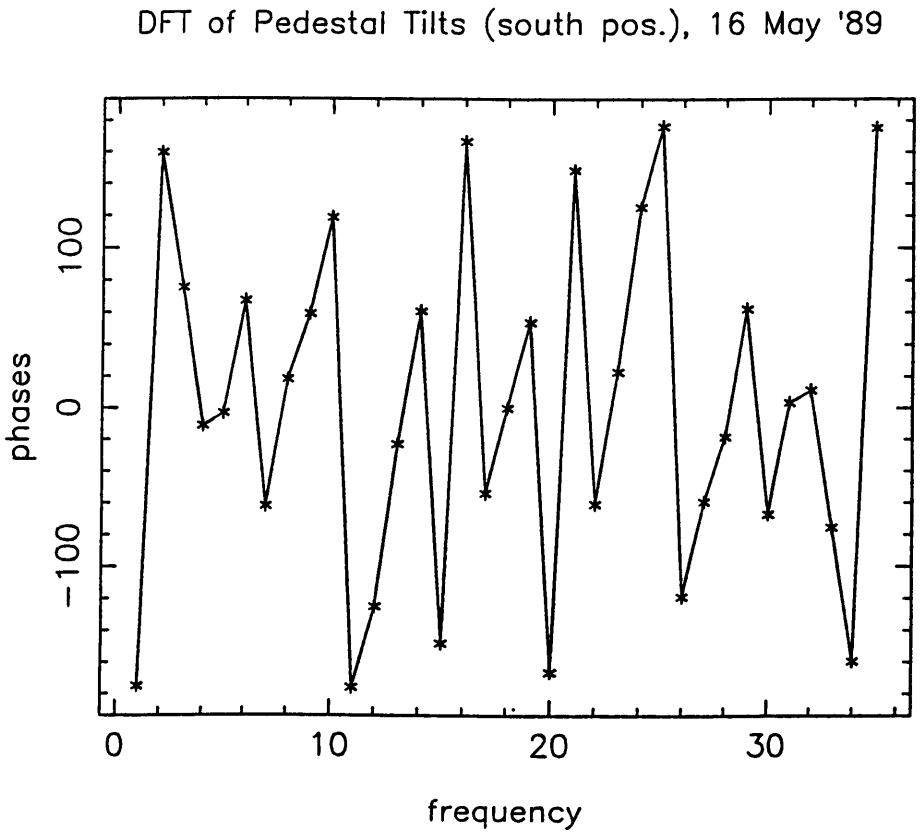
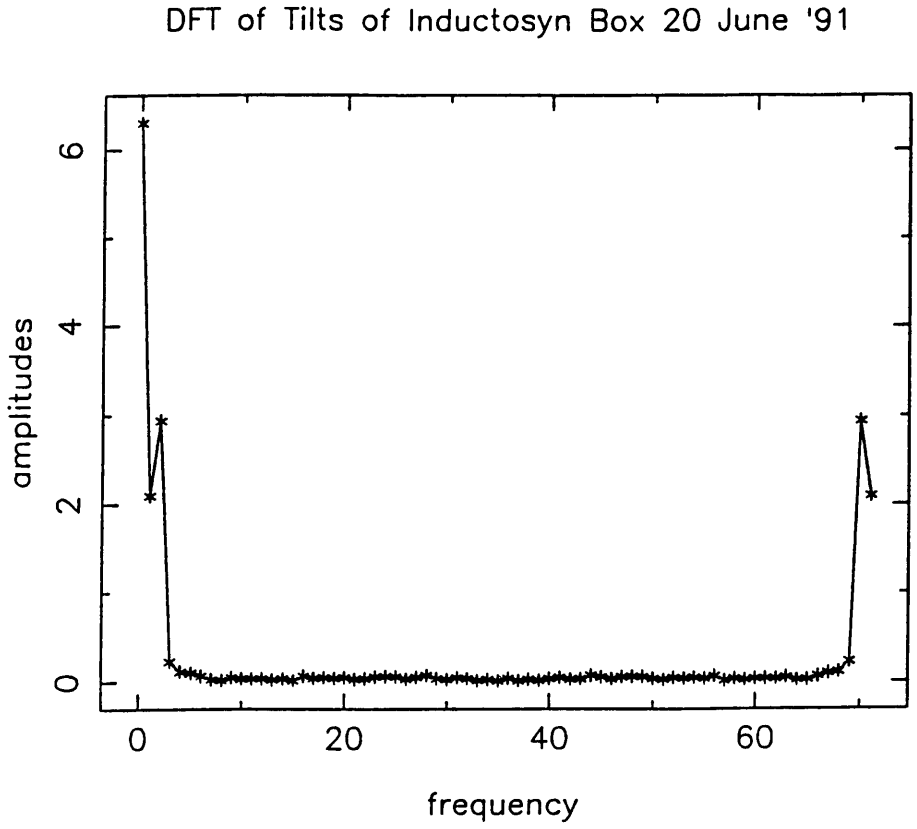

DFT of Pedestal Tilts (south pos.), 16 May '89
phases
-100
0
100
0
10
20
30
frequency

FIG. 7


DFT of Tilts of Inductosyn Box 20 June '91
amplitudes
0
2
4
6
0
20
40
60
frequency

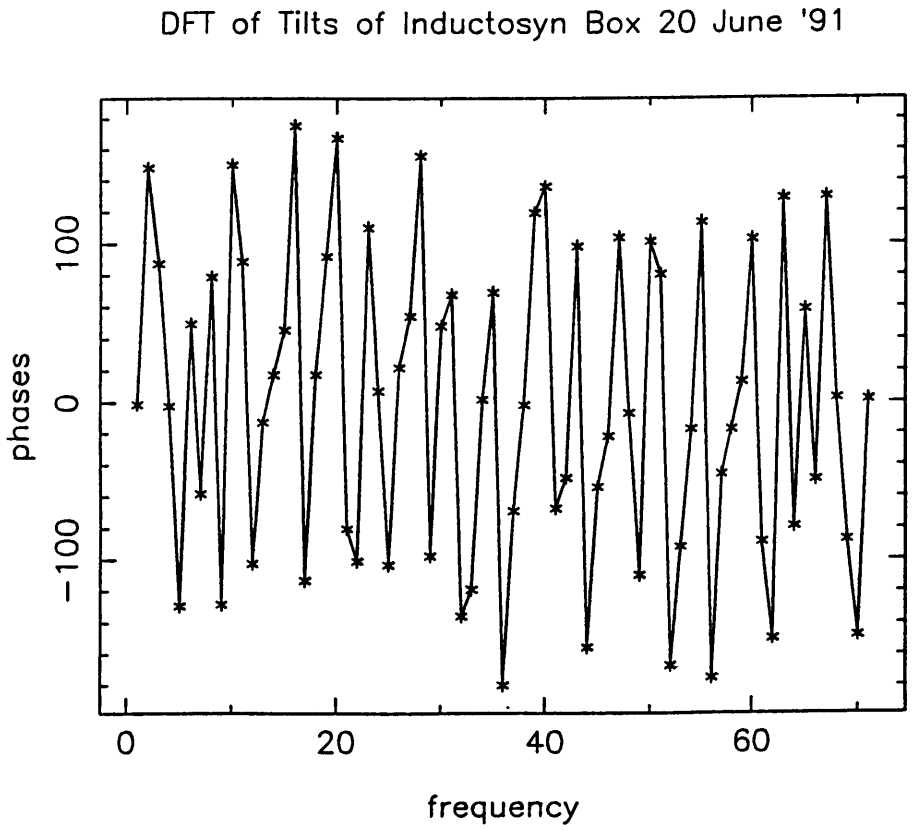
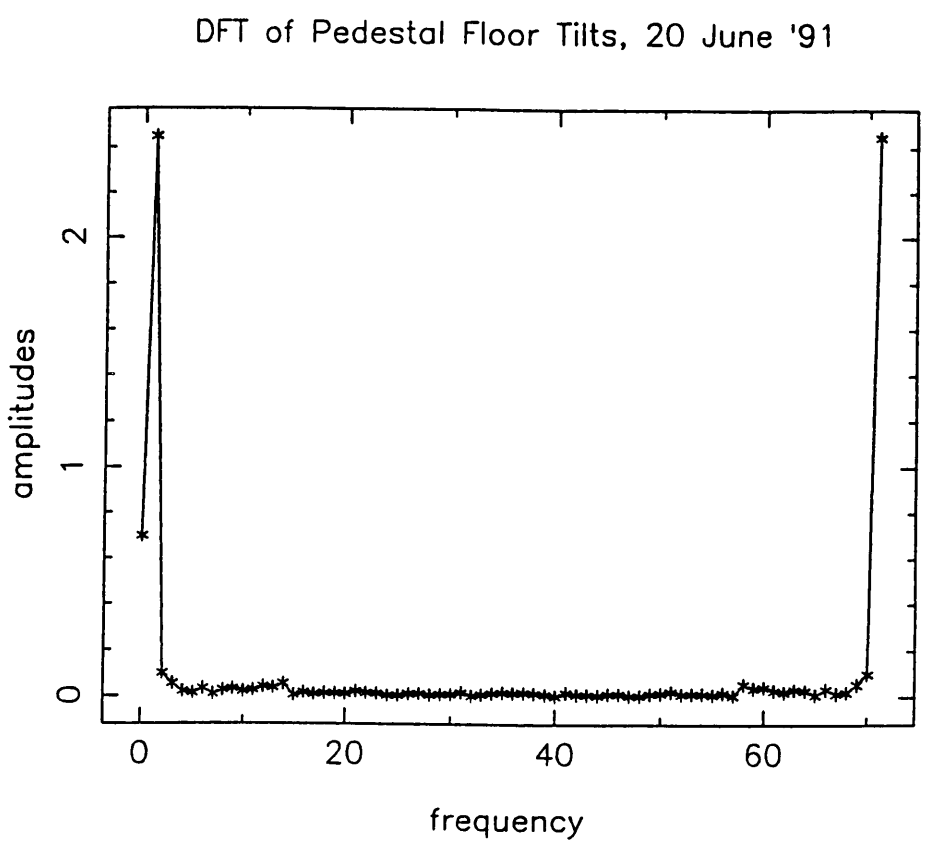

DFT of Tilts of Inductosyn Box 20 June '91
phases
−100
0
100
0
20
40
60
frequency

FIG. 8


DFT of Pedestal Floor Tilts, 20 June '91
amplitudes
0
1
2
0
20
40
60
frequency

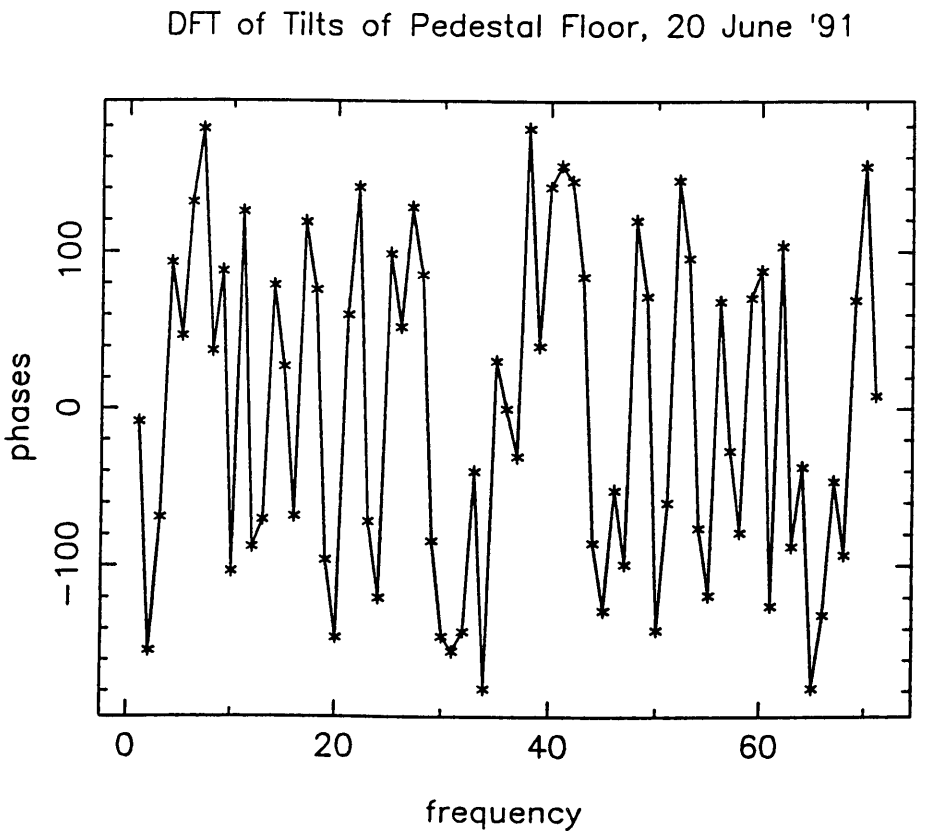
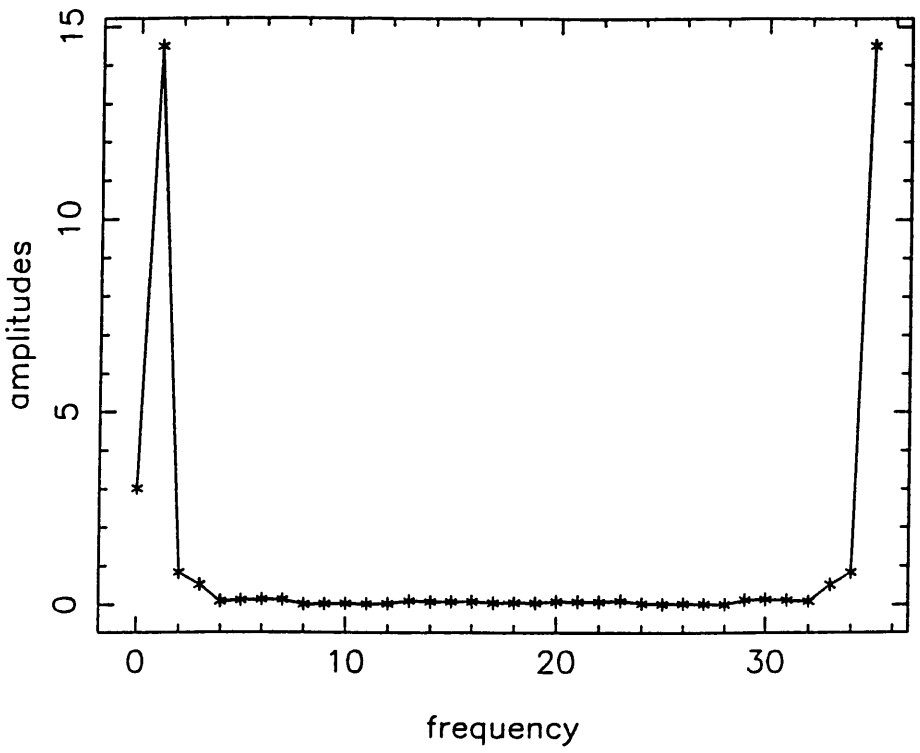

DFT of Tilts of Pedestal Floor, 20 June '91
phases
-100
0
100
0
20
40
60
frequency

FIG. 9


DFT of East Yoke Arm Tilts (perp. pos.) 15 May '91
amplitudes
0
5
10
15
0
10
20
30
frequency

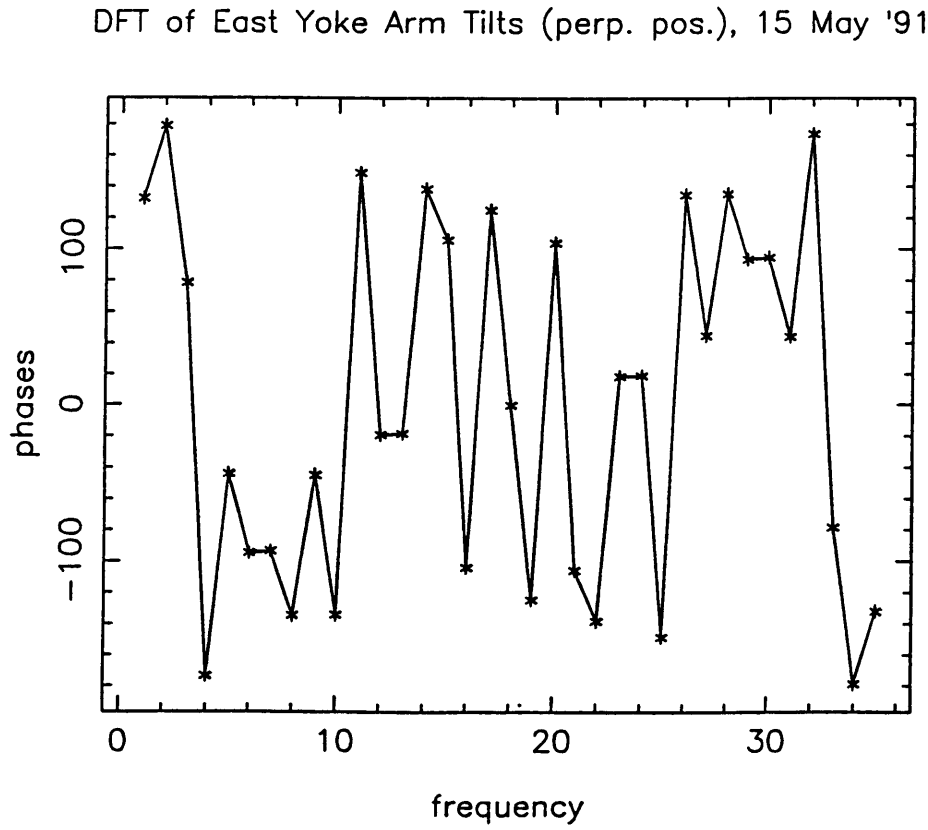
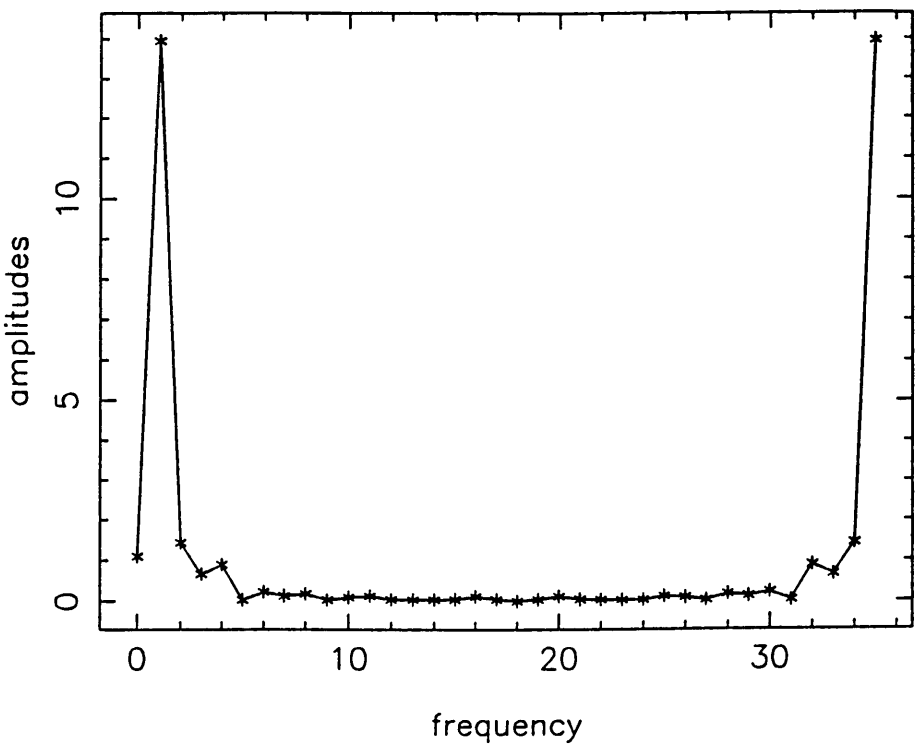

DFT of East Yoke Arm Tilts (perp. pos.), 15 May '91
phases
-100
0
100
0
10
20
30
frequency

FIG. 10


DFT of East Yoke Arm Tilts (parallel pos.), 15 May '91
amplitudes
0
5
10
0
10
20
30
frequency

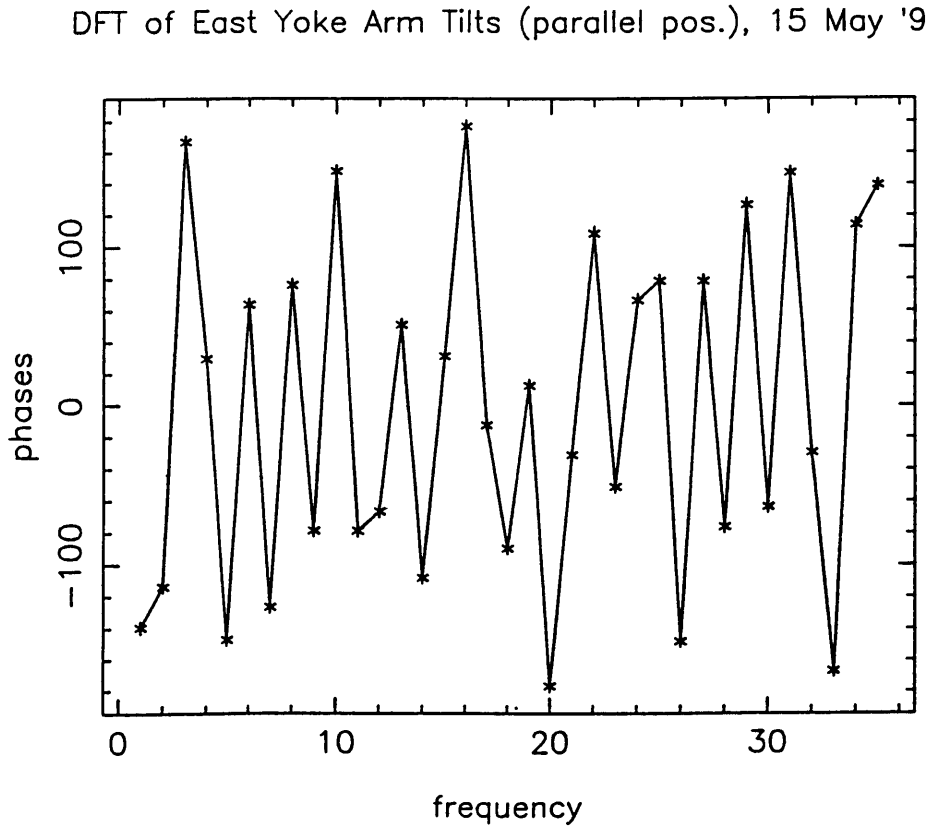

DFT of East Yoke Arm Tilts (parallel pos.), 15 May '91
phases
-100
0
100
0
10
20
30
frequency

FIG. 11

DISTRIBUTION:

12 m Memo Series File

Tucson:
- D. Chase
- D. Emerson
- T. Folkers
- R. Freund
- M. Gordon
- J. Kingsley
- J. Lamb
- J. Payne

Charlottesville:
- R. Brown
- L. King

Socorro:
- C. Janes
- P. Napier
- C. Wade

Others:
- B. L. Ulich (Kaman Aerospace)