

()

:

:

1. : ()

2. : 1999. 1 1999. 12.

3. :

4.

가.

		1	2	3	4	5	6	7	8	9	10	11	12	
0	S/W													
0														
-														
-														
0														
-														
-														
-														
0														
0														
(%)		25%			25%			25%			25%			

.

1)

2)

3)

4)

5.

1)

2)

3)

S/W

6.

o

,

o

o

o

o

7.

.

8.

.

SUMMARY

As Information and communication have rapidly developed, the development of basic element of antenna have been developed from the advanced countries, also the techniques of antenna measurements.

In U.S.A, it starts to study the near-field measurement in 1950's.

Its techniques was used commonly in 1960's. So, we investigated the near-field antenna measurement of brief theory and processing, we make the code of the far-field transform program. In future, When we have built the near-field measurement system, it will be able to be used.

1

2

1

2

3

4

5

6

7

3

1

2

3

4

.....

.....

1

19 Maxwell 가 1887
Hertz 가 . 14
Marconi가
.
.
.
가 가 ,
.
 ,
 ,
 ,
.
1950
가 (Georgia Institute of
Technology) (National Bureau of Standards)
.
.
.

2

1

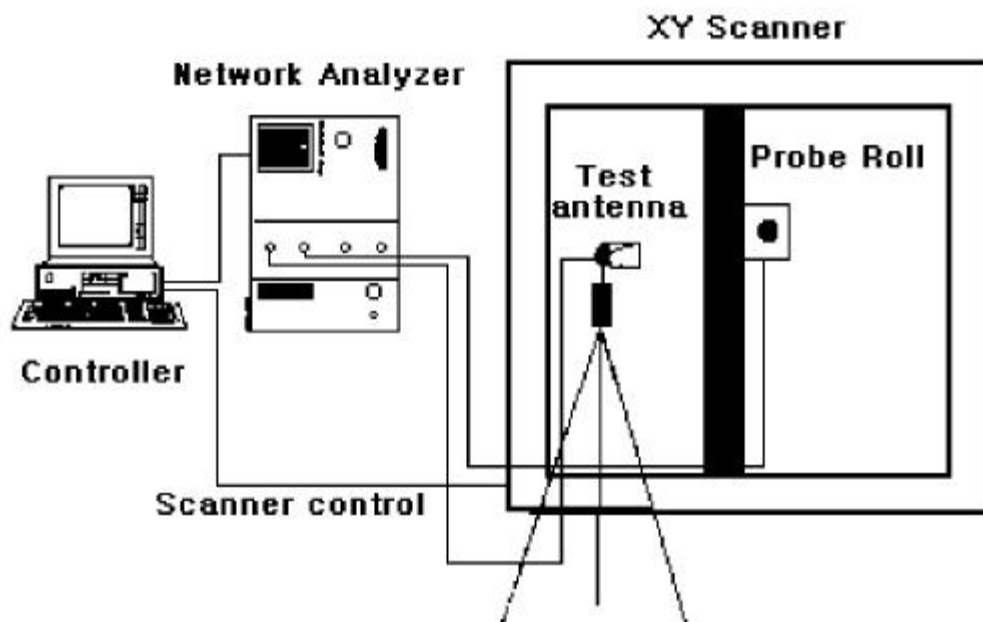
Field

Spherical

Planar, Cylindrical,

() (angular spectrum)
(angular spectrum)

(1)



1

가 가
(reactive)
(conductive surface)
(Fresnel)
(Fraunhofer)

1

22.5

$R = 2D^2/$

D :
(1)

	Evanescent	Near- field	Far- field
Near limit	0	3	$2D^2/$
Far limit	3	$2D^2/$	
Power decay	R-n	1	R-2
E and H orthogonal	no	yes	yes
$Z_0 = 377$	no	yes	yes

15.24

4827.9

22.5

0.01

3.6

,

6

가 .

•

가

•

.

•

.

.

3

.

2

•

(phase front)

.

가 가 .

,

(angular spectrum)

,

.

,

(probe)

,

가

가 .

•

•

,

(synthetic aperture phased array antenna)

,
 . (aperture synthetic)
 .
 (2)

Near- field term	Doppler beam forming	Aperture synthetic
Spatial frequency	doppler frequency	phase tilt
Spatial filter	doppler filter	beam former
Far- field transform	doppler beam former	beam former
Aliasing, undersampling	ambiguity	grating lobes
Near- field	focused	focused
Probe antenna	antenna	element
Probe pattern	antenna pattern	element
Probe correction	deconvolution	array factor
K space		UV plane

3 (Fourier Optics)

- (Optics) .
- 가 . $1 = 360^\circ = 2$
radians = 1 cycle.
- (Phase front) . ,
- (Spherical phase front) .
- (phase front) .
(phase front)
- .
- .
- 0 (field)

0 . , 가

0

○ (probe)

(phase front)

0

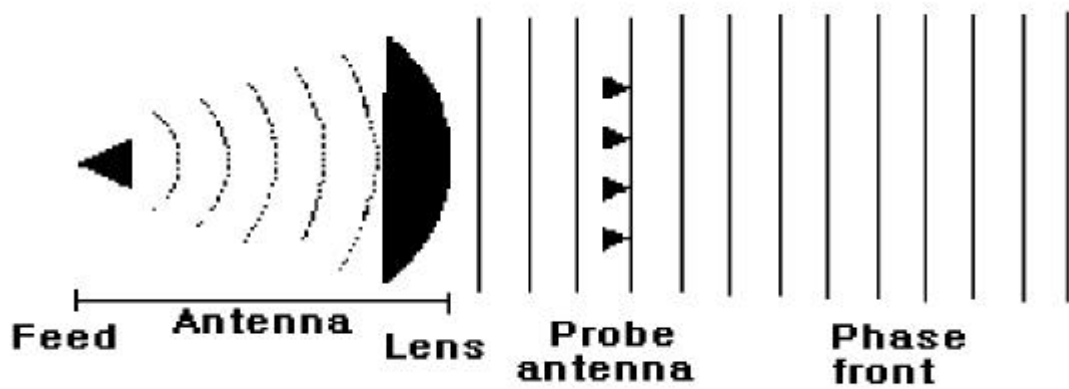
0

가

가

가 . 가 2.4

(2) An idealized antenna with a planar front and four antennas.



0 . 2.4

$$360^\circ \quad 1 \quad .$$

가

가

가

(probe)

(phase front)

. 2 4

4

. 4

가 0°

가

180°

.

4

1

.

4

가

N

N

.

,

(Square Wave)

.

,

(Real)

가

(Imaginary)

.

5

(Phasor)

(Complex value)

(Phasor)

.

$$E = I + Qi$$

$I =$ (in- phase)

$Q = 90$

$i = -1$

(Phasor)

$$|E| = \sqrt{I^2 + Q^2}$$

(Phasor)

.

$$\arg(E) = \arctan(Q/I)$$

(Phasor)

(Phasor)

(Phasor)

(Euler's identity)

$$A = e^{j\theta}$$

$$I^2 + Q^2 = 2$$

(E)

$$I' = I \cos \theta + Q \sin \theta$$

$$Q' = Q \cos \theta - I \sin \theta$$

(X)가

(I, iQ)

$$X' = I \cos \theta + Q \sin \theta + i(Q \cos \theta - I \sin \theta)$$

X

(Euler's identity)

$$A = X e^{j\theta} = X / e^{-j\theta}$$

E

n

가

E

n

$$A_{ave} = 1/n \sum_{j=1}^n E(j)$$

n

(Phasor)

n

(Phasor)

n

(Phasor)가

(aperture phase-front)

(Phasor)

(Boresight)

, 가

(equivalent isotropic power)

(Modification)

(spatial)

(Phasor)

$$A\left(K_x\right)=1/n\sum_{j=1}^nE(j)e^{-iK2^{-x(j)}kx}$$

$$A\left(az\right)=1/n\sum_{j=1}^nE(j)e^{-ik2^{-x(j)}\sin(az)}$$

$$\begin{aligned} & ,\; E(j)=j \\ & X(j)=j\qquad\qquad\qquad (\qquad\quad) \\ & K_x=\text{spatial}\qquad\qquad\qquad (\text{cycle}/\quad)=\sin(az) \\ & K=2^{-\quad}/\quad= \\ & i=-1 \\ & n=\end{aligned}$$

$$\begin{aligned} & 2 \\ & K_x,K_y,K_z\qquad\qquad\qquad 3\qquad\qquad\qquad,\;K_x\qquad\qquad\qquad K_y \\ & K_z\qquad\qquad\qquad\cdot\;K_x,K_y,K_z\qquad\qquad\qquad u,v,w \\ & \qquad\qquad\qquad\cdot\qquad\qquad\qquad(\qquad\quad) \\ & 1\qquad\qquad\qquad/\qquad\qquad\qquad\cdot\end{aligned}$$

RF

2

spatial

“K space”

가

$$A\left(K_x,K_y\right)$$

$$\left(K_x,K_y\right)$$

$$A(K_x,K_y)=1/n\sum_{j=1}^nE(j)e^{-jK(x(j)K_x+y(j)K_y)}$$

$$, \quad E(j) = \mathbf{j}$$

$$X(j) = \mathbf{j} \qquad \qquad \qquad \mathbf{x}$$

$$Y(j) = \mathbf{j} \qquad \qquad \qquad \mathbf{y}$$

$$K=2\div=$$

$$K_x= \hspace{10em} (cycles/ \hspace{0.5em})$$

$$K_y= \hspace{10em} (cycles/ \hspace{0.5em})$$

가 2

$$F(K_x,K_y)=\int\int_0^2f(x,y)e^{-j(K_xx+K_yy)}dxdy$$

$$F(K_x,K_y)=\int\int f(x,y)e^{-j(K_xx+K_yy)}dxdy$$

$$f(x,y)=\int\int F(K_x,K_y)e^{j(k_x x+k_y y)}dK_xdK_y$$

$$, \quad x = x \quad (\hspace{1em})$$

$$y = y \quad (\hspace{1em})$$

$$K=2\div=$$

$$K_x= \hspace{10em} (cycle/ \hspace{0.5em})$$

$$K_y= \hspace{10em} (cycles/ \hspace{0.5em})$$

(raster) 가

2

K

(remapping) K

6 (Doppler beam forming)

가 .

가

(f) RF (f), (v),
() 가 :

$$(f) = f v / c \sin$$

,

(v)

$$f= v \sin$$

11.803GHz

12 가 3 가

10.39 Hz

가 60

가 10.39 Hz 60

가

AUT flies

가

가

3.

()	$f = 12 \sin$ (Hz)	
90	12.00 Hz	
60	10.39 Hz	
30	6.00 Hz	
0	0.00 Hz	
- 30	- 6.00 Hz	
- 60	- 10.39 Hz	
- 90	- 12.00 Hz	

가 .

.

가

가 2

2 . 2

.

.

.

A ()

A (S)

A ()= A [arcsin(S)]

S = f/ (cycles/)

f= (Hz)

(arcsine)

.

.

.

.

AUT

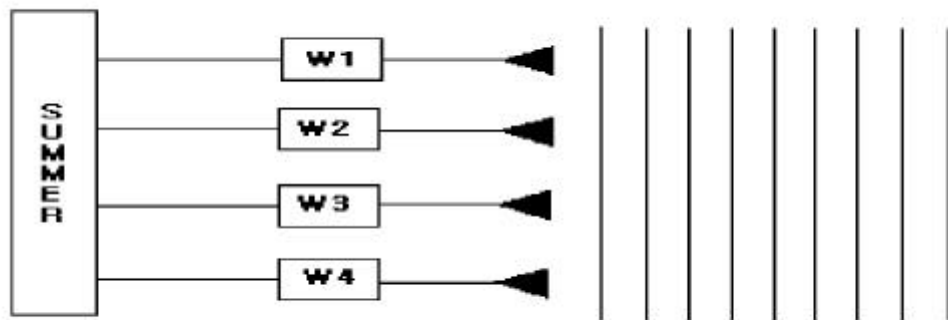
(IF)

(4) Source and Aperture Characteristics of Test Ranges

Test Range	Source	Aperture
Far- field range	distant	real
Compact range	collimated collimated	real(parabolic) real(phased array)
Near- field range	collimated	synthetic(phased array)

on- axis 가 .(3)

가 . 가 180



(3) A phased array antenna illuminated by an on- axis plane wave.

3

dimensions), (spatial filter), (scan pattern

1

- (scan shape : planar, cylindrical, spherical)
- (scan dimensions : height, width, radius, etc)
- (probe antenna specification : type, gain, polarization, etc)
- (scan pattern : raster, starburst, spiral, random)
- (scan parameters : sampling spacing)

- (phase front) Nyquist

(grating lobe)

- 가

. ,
 .
 .
 .
 - 가 ,
 . (grating
 lobe)
 .
 .
 - 가
 .
 . 2
 . (starburst pattern) .

1 (Synthetic Aperture surface shape)

,
 ,
 .
 .
 , , .
 (5) Near- field Scan Surface Comparison

Parameter	Planar	Cylindrical	Spherical
High- gain antenna	excellent	good	good
Low- gain antenna	poor	poor	excellent
Antenna feed elements	good	good	excellent
Stationary AUT	yes	possible	possible
Zero gravity simulation	excellent	poor	variable
Alignment ease	simple	difficult	difficult
Transform	simple	moderate	moderate
Probe correction	simple	complex	complex
Speed	fast	slow	slow

가

.
o

.(RF 가

)

$$\text{scan height} = D + P + 2Z \tan$$

, D=

P=

Z= -

=

가

.

$$\text{scan angle} = \min\{2[\text{arctan}(D/2Z)], 360^\circ\}$$

o RF 가 가 가 가

가 25 45dB .

o (overscanning)

.

가 . 가

(Spatial step function: scan truncation)

(Gibb's ripple)

2.

가 .

o (evanescent) 가 /2 + 1

가 .

가

.

.

- 가 2
- 가 1 2 .
- 가 ,
(Aliasing) .
- (Oversampling) 가
가 .

3.

- (aliasing)
 - (SNR)
 -
 -
- 가. (Open-ended waveguide)
- .
- (Loop antenna)
- (High gain antenna)
- , , (LP) ,

. (Monopulse antenna)

. (Synthetic probe antenna)

Z

/4

가

가

4. (Scan Pattern)

, ,
,
,
, . 가

(Grid)

(Raster)

가

180

가 /2+1

. RF

6

$$(\text{probe gain}) = 0.5 \left[\frac{1}{\tan(M/1.03)} \right]^2$$

$$(\text{scan width}) = D + P + 2Z \tan(M)$$

$$(\text{sample delta}) = 0.5(n/n+1) \frac{1}{\sin(M)}$$

$$(\text{number of samples/ray}) = \text{sample delta} / \text{scan width} = n$$

D=

M=

P=

Z=

=

(6) Representative antenna and starburst scan parameters

Antenna	Frequency (GHz)	Diameter (ft)	Maximum far- field	AUT gain(dB)	Probe gain(dB)	Z distance (ft)	Scan width(ft)	#cuts	Sample delta(in)	Samples /cut
SGH	10.7	0.25	45 °	17	6	0.41	0.45	8	0.78	10
LANDSAT	15.00	6	5 °	47	25	3	7.1	8	4.5	22
ACTS	29.75	9	5 °	57	25	5	10.9	16	2.27	60
TDRSS SA	15.00	15	5 °	56	25	5	17.4	16	4.51	48
STDN	2.30	210	5 °	61	25	50	240	16	29.4	100
Arecibo	430.00	1000	5 °	59	25	250	1200	16	157	92

2

가

가

1

(Raster Far- field Transform)

가

(fast Fourier transform:FFT)

(7) Transformation techniques compared

	2- D FFT	FDFT	DFT
Speed	fast	medium	slow
Round- off errors	good	good	poor
Real time	no	possible	no
Multiplexed beams	yes	yes	yes
Vectorization code	yes	yes	yes
x,y,z error compensation	interpolation	partial	yes
Angle output	2- D interpolation	1- D interpolation	yes
Plane polar	2- D interpolation	yes	yes
Zoom area	2- D interpolation	partial	yes

○ 가 (arcsine) : $f(\)=f[\arcsine(kx)]$
 (mapping) .

:
:

(Discrete fourier transform)
 (Factored discrete fourier transform) . 7

2. (Plane Polar Far-field transform)

, 가 /

(8) Plane Polar Transform Methods Compared

	Fourier- Bessel	Interpolation and FFT	FDFT
Speed	medium	fast	medium
Complexity	difficult	medium	simple
Real time	no	no	possible
Multiplexed beams	marginal	yes	yes
Vectorization	no	yes	yes

가.

angular spectrum by DFT) (Plane polar

K A(Kx,Ky)

$$A(kx, ky) = 1/nw \sum_{j=0}^n E(j)W(j)R(j)$$

$$W(j) = e^{-k[ksub{xx}(j) + k_y y(j) + k_z z(j)]}$$

$$K_z = \begin{cases} \sqrt{1 - k_x^2 - k_y^2}, & \text{for } k_x^2 + k_y^2 < 1 \\ 0, & \text{for } k_x^2 + k_y^2 \geq 1 \end{cases}$$

$$\begin{aligned} x(j) &= j \quad x \\ y(j) &= \quad \text{“} \quad y \quad \text{”} \\ z(j) &= \quad \text{”} \quad z \quad \text{”} \end{aligned}$$

$$E(j) = j$$

$$R(j) = \text{가}$$

$$k=2 \quad / \quad =$$

$$n =$$

$$w =$$

$$k_x =$$

$$k_y =$$

$$k_z =$$

$$R(j) = w \quad 1 \quad , \quad z(j) \quad 0 \quad k_z \quad .$$

$$R(j) \quad , \quad \text{가} \quad \text{가} \quad \text{가} \quad .$$

3.2

가

$$R(j)$$

$$R(j) = \left[\frac{1}{2} (r + \frac{1}{2}) \right]^2 / n - \left[\frac{1}{2} (r - \frac{1}{2}) \right]^2 / n$$

, r=

=

n=

,

.

$$R(j) = \frac{1}{2} r^2 / n$$

, 가 .

$$R(j) = \frac{1}{2} \left(\frac{1}{2} \right)^2 / n$$

(w)

가

가 .

$$w = \sum_{j=0}^n R(j)$$

.

kx, ky kz

kx , ky

kz . kx,ky kz u,v,w .

kx, ky kz z

z . +x .

kx=sin cos

ky=sin sin

kz=cos

kx,ky kz (elevation) (azimuth) .

kx=sin(az)cos(el)

ky=sin(el)

kz=cos(az)cos(el)

=arccos[cos(el)cos(az)]=arccos(kz)

=arctan[tan(el)/sin(az)]=arctan(ky/kz)

$$e_l = \arcsin[\sin(\theta) \sin(\phi)] = \arcsin(ky)$$

$$, \quad W(j) = e^{-ik[\cos(\epsilon l)[\sin(az)x(j) + \cos(az)z(j)] + \sin(\epsilon l)y(j)}$$

$$y(j)=j \quad y$$

$$E(j)=j$$

$$k=2 \quad / \quad =$$

$$W =$$

$$kx =$$

$$ky =$$

$$kz =$$

$$z = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_j \\ \vdots \\ z_n \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_j \\ \vdots \\ x_n \end{pmatrix}, \quad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_j \\ \vdots \\ y_n \end{pmatrix}$$

2

3

$$e^{i(\varphi_1 + \varphi_2)} = e^{i\varphi_1} e^{i\varphi_2}$$

$$A\left(az,el\right)=1/nw\sum_{j=0}^nW_1(j)W_2(j)$$

,

$$\begin{aligned} W_1(j) &= e^{-iK[\sin(el)y(j)]}E(j)R(j) \\ W_2(j) &= e^{-iK[[\sin(az)x(j)]+[\cos(az)z(j)]]} \end{aligned}$$

$$x(j)=j \hspace{10em} x$$

$$y(j)=j \hspace{10em} y$$

$$z(j)=j \hspace{10em} z$$

$$E(j)=j$$

$$R(j)=j \hspace{2em} \text{가}$$

$$K=2 \hspace{1em} / \hspace{1em} =$$

$$n=$$

$$w=$$

.

.

o

.

$$Kx=\sin(az)$$

$$Ky=\sin(el)$$

$$2 \hspace{10em} 0$$

.

o

x,y

.

.

o

, 가

.

o

x 가

.

o

y 가

.

o

가

.

가 y

5 . 2 .
 o .

3 (Probe Compensation)

가 , .
 .

$$\begin{aligned} I_x &= E_p E_{p1} + E_c E_{c1} \\ I_y &= E_p E_{c2} + E_c E_{p2} \end{aligned}$$

, Ix =
 Iy=
 Ep=
 Ec=
 Ep1= 1
 Ec1= 1
 Ep2= 2
 Ec2= 2

Ix Iy 가

. ,

$$\begin{bmatrix} I_x \\ I_y \end{bmatrix} = \begin{bmatrix} E_{p1} & E_{c1} \\ E_{c2} & E_{p2} \end{bmatrix} \begin{bmatrix} E_p \\ E_c \end{bmatrix}$$

Ep Ec

.

$$\begin{bmatrix} E_p \\ E_c \end{bmatrix} = 1/ \begin{bmatrix} E_{p2} & - E_{c1} \\ - E_{c2} & E_{p1} \end{bmatrix} \begin{bmatrix} I_x \\ I_y \end{bmatrix}$$

, = E_{p1}E_{p2}- E_{c1}E_{c2}

가

1, 2 3
 .
 1

가 2 가
가 3
3

4

2

가

가

near- field

가

near- field

Outdoor

compact

가

- [1] Dan Slanter, Near-field antenna measurement, Artech house, 1991
- [2] Gary E. Evans, Antenna measurement Techniques, Artech house, 1990
- [3] IEEE, IEEE Standard test procedures for antenna, Wiley & Sons, 1979
- [4] Ramaon C. Braird, Allen C. Newell, Carl F. Stubenrauch, A brief history of Near-field measurements of Antennas at National Bureau of Standards, IEEE, 1988
- [5] Edward B. Joy, A brief history of the Development of the Near-field measurement technique at the Georgia Institute of Technology, IEEE, 1988

\$cds on

\$files 2,7

\$ema/buffer/

PROGRAM NFFT (INPUT)

cc

c c

c This program reduces two-dimensional near-field data. c

c Written by G.H. Lee and J.S. Park c

c c

c Program last revised: 20 OCT 1999 c

c c

c References for this program(especially the probe c

c correction section): c

cc

COMPLEX SDATA(4096)

COMPLEX DATA(4096,4096),DATA2(4096,4096),BFILT(4096,4096)

EMA DATA, DATA2, DUMMY, BFILT, SDATA

COMPLEX CJ,DUMMY,AO

CHARACTER*80 TITLE, CANS,TEMP, CTIT, CTIT2

CHARACTER*15 INPUT, COFILE, XFILE, FNAME, FNAME2

CHARACTER*1 CFILT

INTEGER DBUFF(15)

LOGICAL REPEAT

LOGICAL SINGLE

COMMON /PARAM/RSCAN(7), CAXIS, POL, TITLE, NAME, IDATE(3), ITIME(3)

COMMON /MINMAX/ AMIN, AMAX, PMIN, PMAX, JMAX, IMAX

COMMON /BUFFER/ABUF(4096), PBUF(4096), IBUF

COMMON /USER/ IWRITE, IREAD

COMMON /WVGE/ S, B, AKO

COMMON /LIMIT/ NXO, NX1, NYO, NY1

COMMON /TRANS/TX,TY,TZ,FILTER,\$XINC,\$YINC

COMMON /RECBUFF/LBUF(8200)

INTEGER*4 TIME0,TIME1,TIME2,TIME3,TIME4,TIME5

INTEGER*4 ElapsedTime

CHARACTER NAME*15,CAXIS*1,POL*8,COPOL*8,XPOL*8,BELL*1

CALL ResetTimer

C Unit number for files:

C

C Unit2 - Aperture data, 1st probe rotation

C Unit3 - Aperture data, 2nd probe rotation

C Unit4 - Spectrum data, 1st probe rotation

C Unit5 - Spectrum data, 2nd probe rotation

C Unit6 - Output file for debugging information

C Unit8 - Pattern data for probe correction(1st rotation)

C Unit9 - Pattern data for probe correction(2nd rotation)

C Unit11 - Input file for unattended run

C Unit13 - Output file for aperture Blackman filter

C Unit14 - Output file for spectral blackman filter

OPEN (UNIT=6, FILE='Output_junk')

IF (INPUT.EQ.'1') THEN

IREAD=1

IWRITE=1

ELSE

IREAD=11

OPEN (UNIT=11, FILE=INPUT)

IWRITE=6

END IF

BELL=CHAR(7)

PI=ACOS(-1.)

CJ=(0.,1.)

DR=PI/180

```

RD=180./PI

CALL DateTime(IDATE,ITIME)
CALL FTIME(DBUFF)
CALL SWIPE
WRITE (1,'(3A1)') CHAR(10),CHAR(10),CHAR(10)
WRITE (1,4)
WRITE (1,'(//,20X,15A2)') DBUFF
WRITE (6,5) (IDATE(I),I=1,3),(ITIME(I),I=1,3)
4  FORMAT ( 20X, '***** PROGRAM NFFT *****' )
5  FORMAT ( '**** PROGRAM NFFT **** "1X,2(I2,')/,I2,I4,2(':',I2)

WRITE (1,97)
WRITE (1,*) 'Default response are shown in parentheses. When' ' ' a choice is'
WRITE (1,*) ' displayed, the first response is default.'
WRITE (1,*) 'Defaults may be selected with the Return key.'
WRITE (1,97)

99  FORMAT ( A80 ) ! For user input with CANS
98  FORMAT ( A ) ! For use with BELL
97  FORMAT ( /// )

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C Input the test data
C

WRITE (1,98) BELL
WRITE (IWRITE,*) ' 1. How many polarizations will be analyzed ? ',' (1 or 2) '
READ (IREAD,99) CANS
NPOL=1
IF (CANS .EQ. '2') NPOL=2
WRITE (IWRITE,*) ' NPOL = ',NPOL

WRITE (IWRITE,*)
IF (NPOL.EQ.2) THEN
    WRITE (IWRITE,*) '2a. For the parallel pole aperture data -'
ELSE
    WRITE (IWRITE,*) ' 2. For the aperture data to be analyzed -'
END IF
CALL NAMFILE(2,0)
COFILE=NAME
WRITE (6,110) NAME
IF (NPOL. EQ.2) THEN
    WRITE (IWRITE,*) '2b. For the cross pole aperture data -'
    CALL NAMFILE(3,0)
    XFILE=NAME
    WRITE (6,110) NAME
END IF

CALL HEADREAD(2,IRDAT)
TEMP=TITLE
COPOL=POL
NX=INT(RSCAN(3))
NY=INT(RSCAN(6))
CANS=CAXIS

IF (NPOL.EQ.2) THEN
    CALL HEADREAD(3,IRDAT)
    IF ( (NX.NE.RSCAN(3)) .OR. (NY.NE.RSCAN(6)) .OR. (CANS.NE.CAXIS) ) THEN
        WRITE (IWRITE,*) '** File mismatch - program aborted **'
        STOP
    END IF
    XPOL=POL
END IF

WRITE (6,112) TEMP
IF (NPOL.EQ.2) WRITE (6,112) TITLE

NXO=1

```

```

NY0=1
NX1=NX
NY1=NY
WRITE (IWRITE,*)
WRITE (IWRITE,*) ' 3. Enter row numbers for starting, ending X:','(1,'NX,')'
READ (IREAD,99) CANS
IF (CANS .GT. ' ') READ (CANS,*) NX0,NX1

WRITE (IWRITE,*)
WRITE (IWRITE,*) ' 4. Enter row numbers for starting, ending Y:','(1,'NY0','NY1,')'
READ (IREAD,99) CANS
IF (CANS .GT. ' ') READ (CANS,*) NY0, NY1

IXINC=1
IYINC=1

WRITE (IWRITE,*)
WRITE (IWRITE,*) ' 5. Enter X thinning increment: (1)'
READ (IREAD,99) CANS
IF (CANS .GT. ' ') READ (CANS,*) IXINC

WRITE (IWRITE,*)
WRITE (IWRITE,*) ' 6. Enter Y thinning increment: (1)'
READ (IREAD, 99) CANS
IF (CANS .GT. ' ') READ (CANS,*) IYINC

WRITE (1,*) 'Data set to be analyzed:'
WRITE (1,*) ' X points ',NX0,' through ',NX1,',every', IXINC,'ht point.'
WRITE (1,*) ' Y points ',NY0,' through ',NX1,',every', IYINC,'th point.'

Time0 = ElapsedTime()

MX = 1 + (NX1-NX0)/IXINC
MY = 1 + (NY1-NY0)/IYINC
AMIN = 100.
AMAX = - 100.

CALL ARRAY_FILL(DATA, NX0, NY0, MX, IXINC, IYINC, 2, 1)
IF (NPOL .EQ. 2) THEN
    CALL ARRAY_FILL(DATA2, NX0, NY0, MX, MY, IXINC, IYINC, 3, 2)
END IF

NX = MX
NY = MY
RSCAN(3) = NX
RSCAN(6) = NY
RSCAN(2) = RSCAN(2)*IXINC
RSCAN(5) = RSCAN(5)*IYINC
17  FORMAT(A4)
110  FORMAT(A15)
112  FORMAT(A80/)

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C In this routine, the DATA arrays are one-dimensional. They are
C re-dimensioned in the subroutines for compactness of data storage
C
C In the subroutines, a row represents a scan of constant Y; a
C column, a scan of constant X. (The first subscript represents
C the row number.) In other words, all data is stored in locally
C packed form with the first subscript varying fastest.
C
C NX      Number of pts. per row - the extent of the first index
C NY      Number of rows - the extent of the second index
C Xinc    Spacing of data along the X axis (inches)
C YINC    Spacing of data along the Y axis (inches)
C FREQ    Frequency in GHz

Time1 = ElapsedTime()

```



```

FREQ = RSCAN(7)
ALAM = 11.80283/FREQ      ! Wavelength
AKO = 2. *PI/ALAM         ! Wave Number
XINC = RSCAN(2)
IF (NX.EQ.1) XINC = 2.*PI
YINC =RSCAN(5)
IF (NY.EQ.1) YINC = 2.*PI

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  NORMALIZE NF DATA

      WRITE (1,*) 'Ready to normalize the aperture data.'
      WRITE (1,*)

C  First, get the feed through level for reference

      WRITE (IWRITE,*)
      WRITE (1,98) BELL
      WRITE (IWRITE,*) '7. Enter the reference amplitude and phase,', 'in dB and degrees.'
      WRITE (IWRITE,*) ' ' (Use the feedthrough values if available.'
      WRITE (IWRITE,*) ' ' Default is the maximum amplitude.)'
      READ (IREAD,99) CANS
      IF (CANS .GT. ' ') READ (CANS,*) AMAX,PMAX
      A0 = CMPLX(AMAX,PMAX)

C  Next, translation in wave- number space

      AKX = 0.
      AKY = 0.
      WRITE (IWRITE,*)
      WRITE (IWRITE,*) '8. Enter normalized wave numbers (Kx,Ky) for'
      WRITE (IWRITE,*) ' ' the desired K-space translation: (0.,0.)'
      READ (IREAD, 99) CANS
      IF (CANS .GT. ' ') READ (CANS,*) AKX,AKY

      AKX = AKX*AK0
      AKY = AKY*AK0

      CALL NFNORM (DATA, NX, NY, AKX, AKY)
      IF (NPOL.EQ.2) CALL NFNORM (DATA2, NX, NY, AKX, AKY)

      WRITE (6,290) AMAX, PMAX
290  FORMAT ( 'Near field normalization:', F10.5,'dB, ', F10.5, 'DEG.'/)
C  Pad input for desired resolution enhancement

      CALL TESTP2(NX,ISXP2)
      CALL TESTP2(NY,ISYP2)
      CALL POWRT(NX,NXP2,ISXP2)
      CALL POWRT(NY,NYP2,ISYP2)

      REPEAT = .TRUE.
      DO WHILE (REPEAT)
        REPEAT = .FALSE.
        SNXRES = ALAM / (XINC*NXP2)
        SNYRES = alam / (YINC*NYP2)
        IF (SNXRES .GT. 1.) THEN
          SNXRES = 1.
          WRITE (IWRITE,*)
          WRITE (IWRITE,*) 'WARNING: X scan less than a wavelength', '. Potential error at'
          WRITE (IWRITE,*) ' resolution enhancement. '
          WRITE (IWRITE,*)
        END IF
        IF (SNYRES .GT. 1.) THEN
          SNYRES = 1.
          WRITE (IWRITE,*)
          WRITE (IWRITE,*) 'WARNING: Y scan less than a wavelength', '. Potential error at'
          WRITE (IWRITE,*) ' resolution enhancement.'
          WRITE (IWRITE,*)
        END IF
      END IF
END IF

```

```

ANXRES = ASIN(SNXRES) * RD
ANTRES = ASIN(SNYRES) * RD

WRITE (IWRITE,220) NXP2,SNXRES,ANXRES, NYP2,SNYRES,ANTRES

220  FORMAT (/ '      Dimension      Resolution      Main-beam Angular Res. ' /'
+      ..... 'F8.4,' ..... 'F8.4,' ..... 'F8.4,' deg. ' , /
+' X      'I6,'      'F8.4,'      'F8.4,' deg. ' , /
+' Y      'I6,'      'F8.4,'      'F8.4,' deg. ' //)

      WRITE (IWRITE,*)
      WRITE (1,98)BELL
      WRITE (IWRITE,*)'9a. Would you like increased resolution on'
      WRITE (IWRITE,*) '      the X-axis ?(N/Y)'
      READ (IREAD,99) CANS
      IF (CANS.EQ. 'Y' .OR. CANS.EQ. 'Y') THEN
        NXP2=NXP2*2
        IF (NXP2.GT.4096) NXP2=NXP2/2
        REPEAT = .TRUE.
      END IF

      WRITE (IWRITE,*)
      WRITE (IWRITE,*) ' 9b. Would you like increased resolution on'
      WRITE (IWRITE,*) '      the Y-axis ? (N/Y)'
      READ (IREAD,99) CANS
      IF (CANS.EQ. 'Y' .OR. CANS.EQ. 'Y') THEN
        NYP2=NYP2*2
        IF (NYP2.GT.4096) NYP2 = NYP2/2
        REPEAT = .TRUE.
      END IF

      END DO

      IF (NX.NE.NXP2 .OR. NY.NE.NYP2) THEN
        CALL EXPAND (DATA,NX,NY,NXP2,NYP2)
        IF (NPOL.EQ.2) CALL EXPAND (DATA2,NX,NY,NXP2,NYP2)
      END IF

      WRITE (6,*) ' Old Dimension = ',NX,NY
      WRITE (6,*) ' New Dimension = ',NXP2,NYP2

      NX = NXP2
      NY = NYP2

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C   FFT Section, including resolution enhancement for a sector
C

      WRITE (1,*) 'Ready for the FFT section.'
      WRITE (1,*)

      WRITE (IWRITE,*)
      WRITE (1,98) BELL
      WRITE (IWRITE,*) ' 10. Does this data set contain independent','column or row measurements? (N/Y)'
      READ (IREAD,99) CANS
      SINGLE = .FALSE.
      IF (CANS.EQ.'Y' .OR. CANS.EQ.'Y') SINGLE = .TRUE.

      Time2 = ElapsedTime()

      IF (SINGLE) THEN
        CALL SEPARATE(XINC,YINC,NPOL,NX,NY,DATA,DATA2,CAXIS)
      ELSE
        DA = XINC *YINC / (4. *PI**S)
        CALL FFT2 (1, NX, NY, DA, DATA)
        IF (NPOL.EQ.2) CALL FFT2 (1, NX, NY, DA, DATA2)
      END IF

C   For area factor in FFT (DA) see Kerns, 3.1-3, P.87

```

```

Time3 = ElapsedTime()

SXINC = ALAM / (NX*SXINC)      ! X increment for spectrum data
IF (XINC .EQ. 0) SXINC=0
SYINC = ALAM / (NY*SYINC)      ! Y increment for spectrum data
IF (YINC .EQ. 0) SYINC=0
SX0 = -(NX/2)*SXINC
SY0 = -(NY/2)*SYINC
RSCAN(1) = SX0
RSCAN(2) = SXINC
RSCAN(3) = NX
RSCAN(4) = SY0
RSCAN(5) = SYINC
RSCAN(6) = NY
RSCAN(7) = -FREQ              ! Negative to indicate spectrum data

WRITE (1,98) BELL
WRITE (IWRITE,*)
WRITE (IWRITE,*) ' 11a. Would you like to examine a sector of'
WRITE (IWRITE,*) '      the data with greater resolution? (N/Y)'
READ (IREAD,99) CANS
IF (CANS.EQ.'Y' .OR. CANS.EQ.'Y') THEN
  SXL = -1
  SXU = 1
  WRITE (IWRITE,*) ' 11b. Enter the sector limits for Kx :',' (-1., 1.)'
  READ (IREAD,99) CANS
  IF (CANS .GT. ' ') READ (CANS,*) SXL,SXU
  IF (SXL.GT.SXU) THEN
    SWAP = SXL
    SXL = SXU
    SXU = SWAP
  END IF
  IL = (SXL-SX0)/SXINC+1
  RIU = (SXU-SX0)/SXINC+1.
  IU = RIU
  IF (FLOAT(IU) .LT. RIU) IU = IU+1

  SYL = -1
  SYU = 1
  WRITE (IWRITE,*) ' 11c. Enter the sector limits for KY :',' (-1., 1.)'
  READ (IREAD,99) CANS
  IF (CANS .GT. ' ') READ (CANS,*) SYL,SYU
  IF (SYL.GT.SYU) THEN
    SWAP = SYL
    SYL = SYU
    SYU = SWAP
  END IF
  JL = (SYL-SY0)/SYINC+1
  RJU = (SYU-SY0)/SYINC +1.
  JU = RJU
  IF (FLOAT(JU) .LT. RJU) JU = JU+1

  NXSECT = IU-IL+1
  NYSECT = JU-JL+1
  CALL TESTP2 (NXSECT,ISXP2)
  CALL TESTP2 (NYSECT,ISYP2)
  CALL POWRT (NXSECT,NXSECT,ISXP2)
  CALL POWRT (NYSECT, NYSECT,ISYP2)
  IF (NXSECT.GT.NX) NXSECT = NX
    IF (NYSECT.GT.NY) NYSECT = NY

  IF (NXSECT.GE.NX .AND. NYSECT.GE.NY) THEN
    WRITE (IWRITE,*) '*WARNING: Sector size is the entire',' data set. No resolution '
    WRITE (IWRITE,*) '      enhancement applied. '
  ELSE
    NXP = NXSECT              ! Old sector size
    NYP = NYSECT              ! (power of 2)
    IUP = IU + (NXP - IU + IL - 1)/2
    IF (IUP.GT.NX) IUP = NX
    IF (IUP.LT.NXP) IUP = NXP

```

```

      ILP = IUP - NXP + 1          ! Index of 1st sector point
      SX0 = SX0 + (ILP-1)*SXINC    ! Coord." " " "
      JUP = JU + (NYP - JU + JL - 1)/2
      IF (JUP.GT.NY) JUP = NY
      IF (JUP.LT.NYP) JUP = NYP
      JLP = JUP - NYP + 1        ! Index of 1st sector point
      SY0 = SY0 + (JLP-1)*SYINC  ! Coord." " " "

      DXSECT = SXINC
      DYSECT = SYINC
      XTENT = DXSECT*NXSECT
      YTENT = DYSECT*NYSECT

      REPEAT = .TRUE.
      DO WHILE (REPEAT)
        REPEAT = .FALSE.
        IF (DXSECT .GT. 1.) THEN
          DXSECT = 1.
          WRITE (IWRITE,*)
          WRITE (IWRITE,*) 'WARNING: Kx spacing > 1. 'Potential error at sector enhancement.'
          WRITE (IWRITE,*)
        END IF
        IF (DYSECT .GT. 1.) THEN
          DYSECT = 1.
          WRITE (IWRITE,*)
          WRITE (IWRITE,*) 'WARNING: Ky spacing > 1. 'Potential error at sector enhancement>'
          WRITE (IWRITE,*)
        END IF
        ADXS = ASIN(DXSECT)*RD
        ADYS = ASIN(DYSECT)*RD
        WRITE (IWRITE,220) NXSECT,DXSECT,ADXS, NYSECT,DYSECT,ADYS
      WRITE (IWRITE,*)
      WRITE (IWRITE,*) ' 11d. Would you like increased','resolution on the X-axis ? (N/Y)'
      READ (IREAD,99) CANS
      IF (CANS .EQ.'Y' .OR. CANS .EQ.'Y') THEN
        NXSECT = NXSECT*2
        IF (NXSECT.GT.4096) NXSECT = NXSECT/2
        DXSECT = XTENT/NXSECT
        REPEAT = .TRUE.
      END IF

      WRITE (IWRITE,*)
      WRITE (IWRITE,*) ' 11e. Would you like increased','resolution on the Y-axis ? (N/Y)'
      READ (IREAD,99) CANS
      IF (CANS .EQ.'Y' .OR. CANS .EQ.'Y') THEN
        NYSECT = NYSECT*2
        IF (NYSECT.GT.4096) NYSECT = NYSECT/2
        DYSECT = YTENT/NYSECT
        REPEAT = .TRUE.
      END IF
    END IF
  END DO

  DSA = SXINC * SYINC
  CALL BLOWUP (DATA,NX,NY,NXP,NYP,ILP,JLP,NXSECT,NYSECT,ALAM,DSA)
  IF (NPOL.EQ.2) CALL BLOWUP (DTA2,NX,NY,NXP,NYP,ILP,JLP,NXSECT,NYSECT,ALAM,DSA)
  RSCAN(1) = SX0
  RSCAN(4) = SY0
  RSCAN(2) = SXINC * NXSECT / NXP
  RSCAN(5) = SYINC * NYSECT / NYP
  RSCAN(3) = NXSECT
  RSCAN(6) = NYSECT
  SXINC = RSCAN(2)
  SYINC = RSCAN(5)
  NX = RSCAN(3)
  NY = RSCAN(6)

  WRITE(6,*) 'Results of sector enhancement:'
  WRITE(6,*) 'Old Dimensions = ',NXP,NYP
  WRITE(6,*) 'New Dimensions = ',NXSECT,NYSECT
END IF

```

END IF

C

C PROBE CORRECTION & OUTPUT CONVERSION

C

```
WRITE (1,*) 'Ready for probe correction section.'
WRITE (1,*)
```

```
WRITE (1,98) BELL
WRITE (IWRITE,*)
WRITE (IWRITE,*) 12. What direction is the first polarization?'
WRITE (IWRITE,*) '      Enter angle (degrees) from Y-axis toward'
WRITE (IWRITE,*) '      minus X: (0.)'
READ (IREAD,99) CANS
IF (CANS .EQ. ' ') THEN
    POLY=0.
ELSE
    READ (CANS,*) POLY
END IF
```

```
WRITE (IWRITE,*) 'First polarization at ',POLY,' DEGREES.'
```

```
WRITE (IWRITE,*)
WRITE (IWRITE,*) 13a. Should a probe correction be used? (N/Y)'
READ (IREAD,99) CANS
ICORR=-1
IF (CANS.EQ.'Y' .OR. CANS.EQ.'Y') THEN
    WRITE (IWRITE,*) 13b. Empirical or Theoretical? (E/T)'
    READ (IREAD,99) CANS
    ICORR=1
    IF (CANS.EQ.'T' .OR. CANS.EQ.'T') ICORR=0

    IPRBR = -1
    WRITE (IWRITE,*) 13c. Enter the the probe rotation-'
    WRITE (IWRITE,*) '      1 for X into Y, or'
    WRITE (IWRITE,*) '      -1 for Y into X : (-1)'
    READ (IREAD,99) CANS
    IF (CANS .EQ. '1') IPRBR=1
    WRITE (IWRITE,*) 'Second polarization at',POLY + IPRBR*90,'degrees.'
```

END IF

```
IF (ICORR.EQ.0) THEN
    A = ALAM/1.6
    B = A/2
    WRITE (IWRITE,*) 13d. Enter the probe dimension in inches.'
    WRITE (IWRITE,*) '      Enter large, small dimensions:',('A','B,')
    READ (IREAD,99) CANS
    IF (CANS .GT. ' ') READ (CANS,*) A,B

    WRITE (IWRITE,*)
    IF (ICORR .EQ. 0) THEN
        WRITE (IWRITE,*) 'Correcting for probe size ',A, x ',B.' ""
    ELSE
        WRITE (IWRITE,*) 'Gain calc. for probe size ',A,' x ',B,' ""
    END IF
END IF
```

END IF

```
IF (ICORR.GT.0) THEN
    WRITE (IWRITE,*)
    WRITE (IWRITE,*) 13d. For the probe pattern (1st pole) -'
    CALL NAMFILE(8,0)
    WRITE (6,110) NAME
    CALL HEADREAD(8,IRDAT)
    WRITE (6,112) TITLE
    IF ( (NX.NE.RSCAN(3)) .OR. (NY.NE.RSCAN(6)) ) THEN
        WRITE (IWRITE,*) '** File mismatch - program aborted**'
        STOP
    END IF

    WRITE (IWRITE,*)
```

```

        WRITE (IWRITE,*)' 13e. For the probe pattern (2nd pole) - '
        CALL NAMFILE(9,0)
        WRITE (6,110) NAME
        CALL HEADREAD(9,IRDAT)
        WRITE (6,112) TITLE
        IF ( (NX.NE.RSCAN(3)) .OR. (NY.NE.RSCAN(6)) ) THEN
            WRITE (IWRITE,*)'** File mismatch - program aborted **'
            STOP
        END IF
    END IF

END IF

WRITE (IWRITE,*)
WRITE (IWRITE,*)' 14a. Specify the type of output data desired:'
WRITE (IWRITE,*)
WRITE (IWRITE,*)'      To output the far-field pattern -- '
WRITE (IWRITE,*)'      Enter "Y" for an azimuth?elevation '
WRITE (IWRITE,*)'                  system (conical about the '
WRITE (IWRITE,*)'                  Y-axis) rotated about the '
WRITE (IWRITE,*)'                  Z axis by a specified angle'
WRITE (IWRITE,*)'      Enter "H" for a Huygens system rotated'
WRITE (IWRITE,*)'                  by a specified angle, '
WRITE (IWRITE,*)'      Enter "Z" for a theta/phi system '
WRITE (IWRITE,*)'                  system (conical about the '
WRITE (IWRITE,*)'                  Z-axis) rotated about the '
WRITE (IWRITE,*)'                  Z axis by a specified angle'
WRITE (IWRITE,*)
WRITE (IWRITE,*)'      Or -- '
WRITE (IWRITE,*)'      Enter "A" for a physical translation '
WRITE (IWRITE,*)'                  of the planar aperture data,'
WRITE (IWRITE,*)'      or Return to output the transverse '
WRITE (IWRITE,*)'                  spectrum data'
READ (IREAD,99) CANS
NPOUT=0
NTRANS=0
IF (CANS.EQ.'Y' .OR. CANS.EQ.'y') NPOUT=1
IF (CANS.EQ.'H' .OR. CANS.EQ.'h') NPOUT=2
IF (CANS.EQ.'Z' .OR. CANS.EQ.'z') NPOUT=3
IF (CANS.EQ.'A' .OR. CANS.EQ.'a') NTRANS=1
IF (NPOL.EQ.1) THEN
    WRITE (IWRITE,*)' 14b. Would you like to output both ',polarization? (N/Y)'
    READ (IREAD,99) CANS
    IF (CANS.EQ.'Y' .OR. CANS.EQ.'Y') NPOL=0
    WRITE (IWRITE,*)' Output ',2-'NPOL,',polarization.'
END IF

IF (NTRANS .NE. 0) THEN
    TX=0.
    TY=0.
    TZ=0.
    WRITE (IWRITE,*)' 14c. Enter translation vector components'
    WRITE (IWRITE,*)'      in inches (X, Y, Z) :      (0..0..0)'
    READ (IREAD,99) CANS
    IF (CANS .GT. ' ') READ (CANS,*) TX,TY,TZ

    FILTER=0.
    WRITE (IWRITE,*)' 14d. Enter low-pass filter radius in '
    WRITE (IWRITE,*)'      normalized wave-number units '
    WRITE (IWRITE,*)'      (Retrun for no filter)'
    READ (IREAD,99) CANS
    IF (CANS .GT. ' ') READ (CANS,*) FILTER

    WRITE (IWRITE,*)' Data origin translated to (' ,TX,TY,TZ,')'
    WRITE (IWRITE,*)' Filter applied at Kt = ',FILTER
END IF

POLOUT = POLY

IF (NPOUT.NE.0) THEN
    WRITE (IWRITE,*)
    WRITE (IWRITE,*)' 14e. What direction is the desired output ',polarization? Enter '

```

```

        WRITE (IWRITE,*)      angle (degrees) from Y-axis toward ',minus X:  (',PLOY,')'
        READ (IREAD,99) CANS
        IF (CANS .GT. ' ') READ (CANS,*) POLOUT
        WRITE (IWRITE,*)
        WRITE (IWRITE,*) Output pole referenced to ',POLOUT,'degrees.'
        WRITE (IWRITE,*)
    END IF

    POLY = POLY*DR          ! Convert to radians
    POLOUT = POLOUT*DR

    IF (SINGLE) THEN
        IF (CAXIS.EQ. 'R') THEN
            RSCAN(4) = 0.
            RSCAN(5) = 0.
        ELSE
            RSCAN(1) = 0.
            RSCAN(2) = 0.
        END IF
    END IF

13  FORMAT(A)

        WRITE (IWRITE,*) 15. Do you want to apply a',Blackman filter(N/Y)?'
        READ (IREAD,99) CANS
        IBM=0
C      IF (CANS.EQ.'Y' .OR. CANS.EQ.'Y') IBM=1
        IF (CANS.EQ.'Y' .OR. CANS.EQ.'Y') THEN
            CFILT=' '
            CTIT=' '
            CTIT2=' '
            IBM=1
C          WRITE (IWRITE,*) 15a. Enter output form for filter, S for'
C          WRITE (IWRITE,*)'space domain, W for wave number, B for'
C          WRITE (IWRITE,*)'both, CR for none. '
C          READ (IREAD,13) CFILT
C          IF (CFILT .EQ. 'S' .OR. CFILT .EQ. 'B' .OR. CFILT .EQ. 's' .OR. CFILT .EQ. 'b') THEN
C              WRITE (IWRITE,*) 15b. Give name for spatial filter','output file.'
C              CALL NAMEFILE(13,1)
C              FNAME=NAME
C              WRITE (IWRITE,*) 15c. Default title is ',TITLE
C              WRITE (IWRITE,*) Enter alternate title(cr to default)'
C              READ(IREAD,99) CTIT
C          END IF
C          IF (CFILT .EQ. 'W' .OR. CFILT .EQ. 'B' .OR. CFILT .EQ. 'w' .OR. CFILT .EQ. 'b') THEN
C              WRITE (IWRITE,*) 15b. Give name for wave # filter','output file.'
C              CALL NAMFILE(14,1)
C              FNAME2=NAME
C              WRITE (IWRITE,*) 15c. Default title is ',TITLE
C              WRITE (IWRITE,*) Enter alternate title(CR to default)'
C              READ(IREAD,99) CTIT2
C          END IF
C      END IF

        END IF

    IF (NPOL.NE.1) THEN
        CALL PCORR (DATA, NX, NY, DATA2, NX, NY, ICORR, IPRBR, NPOL, NPOUT, POLY, POLOUT)
        IF (IBM.EQ.1) CALL BLACKMAN (NPOL,ALAM,NX,NY,BFILT,DATA,NX,NY,DATA2,CTIT,CTIT2,CFILT,FNAME,FNAME2)
    ELSE
        CALL PCORR (DATA, NX, NY, DUMMY, 1, 1, ICORR, IPRBR, NPOL, NPOUT, POLY, POLOUT)
        IF (IBM.EQ.1) CALL BLACKMAN (NPOL, ALAM, NX, NY, BFILT, DATA, 1, 1,
DUMMY,CTIT,CTIT2,CFILT,FNAME,FNAME2)
    END IF

    IF (NPOUT .EQ. 1) THEN
    IF (POLOUT.EQ.0) TYHEN
        COPOL = 'El.'
        XPOL = 'Az.'
    ELSE
        WRITE (COPOL,(F4.0, "El")) POLOUT*RD
            ! "Elevation" pole relative to Y-axis
            ! rotated by angle POLOUT

```

```

WRITE (XPOL, '(F4.0,"Az")') POLOUT*RD
                                ! "Azimuth" pole relative to Y-axis
                                ! rotated by angle POLOUT
END IF
ELSE IF (NPOUT.EQ. 2) THEN
WRITE (COPOL, '(F4.0,"HyA")') POLOUT*RD
                                ! Huygens pole "A" relative to Y-axis
                                ! rotated by angle POLOUT
WRITE (XPOL, '(F4.0,"HyB")') POLOUT*RD
                                ! Huygens pole "B" relative to Y-axis
                                ! rotated by angle POLOUT
ELSE IF (NPOUT.EQ. 1) THEN
IF (POLOUT.EQ.0) THEN
COPOL = 'Theta'
XPOL = 'Phi'
ELSE
WRITE (COPOL, '(F4.0,"Th.")') POLOUT*RD
                                ! "Theta" pole relative to Z-axis
                                ! rotated by angle POLOUT
WRITE (XPOL, '(F4.0,"Phi")') POLOUT*RD
                                ! "Phi" pole relative to Z-axis
                                ! rotated by angl POLOUT
END IF
ELSE IF (ICORR.EQ.0 .AND. NPOUT.EQ.0) THEN
COPOL = 'Ver. (Y)'
XPOL = 'Hor.(X)'
END IF

TR = ABS(TX) + ABS(TY) + ABS(TZ) + ABS(FILTER)

IF (TR.NE. 0.) THEN
IF (SINGLE) THEN
CALL SEPTRANS(XINC,YINC,NPOL,NX,NY,DATA,DATA2,CAXIS)
ELSE
CALL TRANSLATE (DATA, NX, NY, TX, TY, TZ, FILTER)
IF (NPOL.EQ.2) CALL TRANSLATE (DATA2,NX,NY,TX,TY,TZ,FILTER)
DSA = SXINC * SYINC*AK0**2
CALL FFT2 (-1, NX, NY, DSA, DATA)
IF (NPOL.EQ.2) CALL FFT2 (-1, NX, NY, DSA, DATA2)
END IF

XINC = ALAM / (NX*SXINC)
IF (SXINC.EQ. 0) XINC=0
YINC = ALAM / (NY*SYINC)
IF (SYINC.EQ. 0) YINC=0
RSCAN(7) = FREQ
RSCAN(1) = -(NX/2) * XINC
RSCAN(2) = XINC
RSCAN(4) = -(NY/2) * YINC
RSCAN(5) = YINC
END IF

IF (RSCAN(2).EQ. 0) RSCAN(2)=RSCAN(5)
                                ! IF 0, ARBITRARILY SET
IF (RSCAN(5).EQ. 0) RSCAN(5)=RSCAN(2)
                                ! INC1=INC2

```

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

```

```

C

```

```

                OUTPUT

```

```

C

```

```

WRITE (1,*)'Ready to output spectrum data files.'

```

```

WRITE (1,*)

```

```

CALL CONVERT (DATA, NX, NY)

```

```

IF (NPOL.NE.1) CALL CONVERT (DATA2, NX, NY)

```

```

POLOUT = POLOUT * RD

```

```

IF (NPOL.EQ.0) THEN

```

```

    XFILE = 'for 2nd pole'

```



```

TITLE = 'Second output polarization.'
END IF

QRITE (1,98) BELL
WRITE (IWRITE,*)
201 FORMAT (///,5x, 'Ready to output results from file',4A)
IF (NPOL.EQ.1) THEN
    WRITE (IWRITE,201) ' ',COFILE
ELSE
    WRITE (IWRITE,201)'s', COFILE,' and',XFILE
END IF
IF (NTRANS.EQ.1) THEN
    IF (NPOL.EQ.1) THEN
202        FORMAT (/ ' 16. This file contains data translated by (' ,3F7.2,')')
        WRITE (IWRITE,202) TX,TY,TZ
        WRITE (IWRITE,*)' Enter data file name:'
        READ(IREAD,98) COFILE
    ELSE
203        FORMAT (/ ' 16',A, '. The',A," file contains ' , a,'- pole','translated data.' )
        WRITE (IWRITE,203)'a', 'first', COPOL
        WRITE (IWRITE,*)' Enter data file name:'
        READ(IREAD,98) COFILE
        WRITE (IWRITE,203) 'b', 'second', XPOL
        READ(IREAD,98) XFILE
    END IF
ELSE IF (NPOUT.EQ.0) THEN
204    FORMAT (/A, ' file contains ',A,'polarized spectrum data. ')
    IF (NPOL.EQ.1) THEN
        WRITE (IWRITE,204)' 16. This', COPOL
        WRITE (IWRITE,*)' Enter data file name:'
        READ(IREAD,98) COFILE
    ELSE
        WRITE (IWRITE,204)' 16a. The first', COPOL
        WRITE (IWRITE,*)'Enter data file name:'
        READ(IREAD,98) COFILE
        WRITE (IWRITE,204)' 16b. The second', XPOL
        WRITE (IWRITE,*)' Enter data file name:'
        READ(IREAD,98) XFILE
    END IF
ELSE
205    FORMAT (/ ' 16',A, ' file contains pattern data which is ',A,' polarized' ,/ '      relative to the Y-axis', ' rotated ',I4,
degrees. ')
    IF (NPOL.EQ.1) THEN
        WRITE (IWRITE,205)' . This', 'elevation', polout
        WRITE (IWRITE,*)'Enter data file name:'
        READ(IREAD,98) COFILE
    ELSE
        WRITE (IWRITE,205)'a. The first', 'azimuth', POLOUT
        WRITE (IWRITE,*)'Enter data file name:'
        READ(IREAD,98) XFILE
    END IF
END IF
WRITE (6,110) COFILE, ' ', XFILE

206    FORMAT (/ ' 17',A, ' The default title for file ',A,, ' is:', //,A80// '      Enter a new title, or RETURN to default: '/')

IF (NPOL.EQ.1) THEN
    WRITE (IWRITE,206)' .', COFILE, TEMP
    READ (IREAD,99) CANS
    IF (CANS .GT. ' ') TEMP = CANS
    WRITE (6,112) TEMP
ELSE
    WRITE (IWRITE,206)'a.', COFILE, TEMP
    READ (IREAD,99) CANS
    IF (CANS .GT. ' ') TEMP = CANS
    WRITE (6,112) TEMP
    WRITE (IWRITE,206)'b.', XFILE, TITLE
    READ (IREAD,99) CANS
    IF (CANS .GT. ' ') TITLE = CANS
    WRITE (6,112) TITLE

```

```

END IF

WRITE (IWRITE,*)' 17. New data file dimensions are (' ,NY,' x ' ,NX;)'
WRITE (IWRITE,*)'Would you like to change the file dimensions', (Y/N)?'
READ (IREAD,99) CANS
IF (CANS .EQ. 'Y' .OR. CANS .EQ. 'y') THEN
    CALL GRIDSET(4096,0,ISTARTX,ISTARTY,MX,MY,NX0,NY0,IXINC,IYINC)
    RSCAN(1)=RSCAN(1) + (ISTARTX)*RSCAN(2)
    RSCAN(2)=RSCAN(2)*IXINC
    RSCAN(3)=MX
    RSCAN(4)=RSCAN(4) + (ISTARTY-1)*RSCAN(5)
    RSCAN(5)=RSCAN(5)*IYINC
    RSCAN(6)=MY
ELSE
    ISTARTX=1
    ISTARTY=1
    NX0=NX
    NY0=NY
    IXINC=1
    IYINC=1
END IF

CALL XYZOPEN(COFILE,4,1)                                !OPEN FILE FOR 1ST POL
IF (NPOL .EQ. 2) THEN
    CALL XYZOPEN(XFILE,5,1)                                !OPEN FILE FOR 2ND POL
END IF

Time4 = ElapsedTime()

CALL DateTime (IDATA,ITIME)

IF (NPOL.NE.1) THEN
    NAME = XFILE
    POL = XPOL
    CALL ARRAY_DUMP (DATA2,NX,NY,NX0,NY0,IXINC,IYINC,ISTARTX,ISTARTY,5)
    CALL HEADWRITE (5,IRDAT)
    WRITE(1,*) 'MAXIMUM FOR CROSS-POL FILE IS',AMAX
END IF

NAME = COFILE
POL = COPOL
TITLE = TEMP
CALL ARRAY_DUMP (DATA,NX,NY,NX0,NY0,IXINC,IYINC,ISTARTX,ISTARTY,4)
CALL HEADWRITE (4,IRDAT)
WRITE(1,*) 'MAXIMUM FOR COPOL FILE IS',AMAX

Time5 = ElapsedTime()

WRITE (6,*)
WRITE (6,*) ' Time to input data: ', TIME1-TIME0,' ms'
WRITE (6,*) ' Condition for FFT      ', TIME2-TIME1,' ms'
WRITE (6,*) ' Perform FFT           ', TIME3-TIME2,' ms'
WRITE (6,*) ' Output data:          ', TIME5-TIME4,' ms'
WRITE (6,*)
WRITE (6,*) ' *** NORMAL TERMINATION ***'
WRITE (1,98) BELL
WRITE (1,*) ' *** NORMAL TERMINATION ***'
END

```

```

$CDS ON
$EMA /BUFFER/
      SUBROUTINE ARRAY_DUMP(CBUF,NX,NY,NX0,NY0,IXINC,IYINC,ISTARTX,ISTARTY,IUNIT)

C      Program last revised: 20 OCT 1999

      CHARACTER  CSCAN*80,CAXIS*1,NAME*15,POL*8
      COMPLEX CBUF(NX,NY)
      COMMON /PARAM/ RSCAN(7), CAXIS,POL,CSCAN,NAME,IDATE(3),ITIME(3)
      COMMON /BUFFER/ ABUF(4096),PBUF(4096),IBUF
      COMMON /MINMAX/ AMIN,AMAX,PMIN,PMAX,MAXY,MAXX
      EMA CBUF

C      SUBOUTINE TO WRITE AMP, PHASE TO DISK FILE

      AMIN = 100.
      AMAX = - 100.
      PMIN = 180.
      PMAX = - 180.
      MAXY = 0
      MAXX = 0

      IBUF=0

      IF (CAXIS.EQ.'Y') THEN
          IROW=1
          DO J=ISTARTX,NX0,IXINC
              IPT=1
              DO I=ISTARTY,NY0,IYINC
                  ABUF(IPT) = REAL(CBUF(J,I))
                  PBUF(IPT) = AIMAG(CBUF(J,I))
                  IPT=IPT+1
              END DO
              CALL WRITE_DATA (IUNIT,IROW,2,2,ABUF,PBUF,IBUF,AMIN,AMAX, PMIN,PMAX,MAXY,MAXX)
              IROW=IROW+1
          END DO
      ELSE
          IROW=1
          DO J=ISTARTY,NY0,IYINC
              IPT=1
              DO I=ISTARTX,NX0,IXINC
                  ABUF(IPT) = REAL (CBUF(I,J))
                  PBUF(IPT) = AIMAG(I,J))
                  IPT=IPT+1
              END DO
              CALL WRITE_DATA (IUNIT,IROW,2,2,ABUF,PBUF,IBUF,AMIN,AMAX,PMIN,PMAX,MAXY,MAXX)
              IROW=IROW+1
          END DO
      END IF

      RETURN
      END

```

```

$CDS ON
$EMA /BUFFER/
      SUBROUTINE ARRAY_FILE(CBUF,NX0,NY0,MX,MY,IXINC,IYINC,IUNIT,IPOL)

C      Program last revised: 20 OCT 1999

      CHARACTER  CSCAN*80,CAXIS*1,NAME*15,POL*80
      COMPLEX CBUF(MX,MY)
      COMMON /PARAM/ RSCAN(7), CAXIS,POL,CSCAN,NAME,IDATE(3),ITIME(3)
      COMMON /BUFFER/ ABUF(4096),PBUF(4096),IBUF
      COMMON /MINMAX/ AMIN,AMAX,PMIN,PMAX,MAXY,MAXX
      EMA CBUF

C      ARRAY_FILL fills the data array in memory from the data file on disk.

      NX=INT(RSCAN(3))
      NY=INT(RSCAN(6))

      IF (CAXIS.EQ.'Y') THEN
        DO I=1,MX
          IROW = NX0 + (I-1)*IXINC
          CALL READ_DATA(IUNIT,IROW,2,2,ABUF,PBUF,IBUF)
          !READFROM FILE
          DO J=1,MY
            JN = NY0 + (J-1)*IYINC
            IF (IPOL.EQ.1) THEN
              IF (ABUF(JN) .GT. AMAX) THEN
                AMAX = ABUF(JN)
                PMAX = PBUF(JN)
                MAXX = I
                MAXY = J
              END IF
            END IF
            IF (ABUF(JN) .LT. AMIN) AMIN=ABUF(JN)
            CBUF(I,J)=CMPLX(ABUF(JN),PBUF(JN))
          END DO
        END DO
      ELSE
        DO J=1,MY
          IROW = NY0 + (J-1)*IYINC
          CALL READ_DATA(IUNIT,IROW,2,2,ABUF,PBUF,IBUF)
          !READ FROM FILE
          DO I=1,MX
            IN = NX0 + (I-1)*IXINC
            IF (IPOL.EQ.1) THEN
              IF (ABUF(IN) .GT. AMAX) THEN
                AMAX = ABUF(IN)
                PMAX = PBUF(IN)
                MAXX = I
                MAXY = J
              END IF
            END IF
            IF (ABUF(IN) .LT. AMIN) AMIN=ABUF(IN)
            CBUF(I,J)=CMPLX(ABUF(IN),PBUF(IN))
          END DO
        END DO
      END IF

      RETURN
      END

```

\$CDS ON

```
SUBROUTINE BLACKMAN(NPOL,ALAM,NX,NY,BFILT,DATA,NX2,NY2,DATA2,
+          CTIT,CTIT2,CANS,FNAME,FNAME2)

CHARACTER CANS*1,C1*1,CAXIS*1,POL*8,NAME*15,CSCAN*80
CHARACTER FNAME*15,FNAME2*15,CTIT*80,CTIT2*80,TEMP*80
COMPLEX CJ,TVAR,BFILT(NX,NY),DATA(NX,NY),DATA2(NX2,NY2)
EMA BFILT,DATA,DATA2
COMMON /PARAM/RSCAN(7),CAXIS,POL,CSCAN,NAME,IDATE(3),ITIME(3)
COMMON /USER/IWRITE,IREAD
```

C

SUBROUTINE TO APPLY BLACKMAN FILTER

```
CJ=(0.,1)
PI=ACOS(- 1.)
RD=180./PI
DR=PI/180.
NUM=NX*NY
DELX=RSCAN(2)
DELY=RSCAN(5)

XINC=ALAM/(DELX*NX)
YINC=ALAM/(DELY*NY)
XMIN=NX*XINC/2
YMIN=NY*YINC/2

KXLPB=PI/XINC
KYLPB=PI/YINC
TX=3*PI/KXLPB
TY=3*PI/KYLPB

AKXMIN=- ALAM/(2*XINC)
AKYMIN=- ALAM/(2*YINC)
AKXDEL=ALAM/(NX*XINC)
AKYDEL=ALAM/(NY*YINC)

HBMAX=0.
DO I=1,NY
    Y=YMIN+(I- 1)*YINC
    ARGY=PI*Y/TY
    DO J=1,NX
        X=XMIN+(J- 1)*XINC
        IF (ABS(X) .GT. TX .OR. ABS(Y) .GT. TY) THEN
            HBXY=0.
        ELSE
            ARGX=PI*X/TX
            HBXY=0.42+.5*COS(ARGX) + 0.08*COS(2*ARGX)
            HBXY=HBXY*(0.42+.5*COS(ARGY) 0.08*COS(2*ARGY))
            HBMAX=AMAX1(HBMAX,HBXY)
        END IF
        BFILT(J,I)=CMPLX(HBXY,0.0)
    END DO
END DO

DO I=1,NY
    DO J=1,NX
        BFILT(J,I)=BFILT(J,I)/HBMAX
    END DO
END DO

DO I=1,NY
    DO J=1,NX
        TVAR=BFILT(J,I)
        CALL POLAR(TVAR,RE,AI)
        IF (RE .LE. 0) THEN
            RE=- 99.
        ELSE
            RE=20*ALOG10(RE)
        END IF
        BFILT(J,I)=CMPLX(RE,AI*RD)
```

```

        END DO
END DO

IF (CANS .EQ. 'S' .OR. CANS .EQ. 'B') THEN
    TEMP=CSCAN
    IF (CTIT .GT. ' ') CSCAN=CTIT
    CALL DUMP_FILTER(BFILT,NX,NY,13,FNAME)
    CSCAN=TEMP
END IF

DO I=1,NY
    DO J=1,NX
        RE=REAL(BFILT(J,I))
        RE=10.*(RE/20.)
        AI=AIMAG(BFILT(J,I))
        BFILT(J,I)=RE*CEXP(CJ*AI*DR)
    END DO
END DO

CALL FFT2(1,NX,NY,1,BFILT)
HBMAX=0.

DO I=1,NY
    DO J=1,NX
        HBMAX=AMAX1(CABS(BFILT(J,I)),HBMAX)
    END DO
END DO

DO I=1,NY
    AKY=AKYMIN + (I-1)*AKYDEL
    DO J=1,NX
        AKX=AKXMIN + (J-1)*AKXDEL
        BFILT(J,I)=BFILT(J,I)/HBMAX
        IF (CABS(BFILT(J,I)) .LT. 0.03162) THEN
            DATA(J,I)=CMPLX(0.0,0.0)
            IF (NPOL .NE. 1) DATA2(J,I)=CMPLX(0.0,0.0)
        ELSE
            DATA(J,I)=DATA(J,I)/BFILT(J,I)
            IF (NPOL .NE. 1) DATA2(J,I)=DATA2(J,I)/BFILT(J,I)
        END IF

        IF ((AKX*AK0)**2+(AKY*AK0)**2 .GT. KYLPB**2+KXLPB**2) THEN
            DATA(J,I)=CMPLX(0.0,0.0)
            IF (NPOL .NE. 1) DATA2(J,I)=CMPLX(0.0,0.0)
        END IF
        TVAR=BFILT(J,I)
        CALL POLAR(TVAR,RE,AI)
        IF (RE .LE. 0) THEN
            RE=-.99.
        ELSE
            RE=20*ALOG10(RE)
        END IF
        BFILT(J,I)=CMPLX(RE,AI*RD)
    END DO
END DO

IF (CANS .EQ. 'W' .OR. CANS .EQ. 'B') THEN
    TEMP=CSCAN
    IF (CTIT2 .GT. ' ') CSCAN=CTIT2
    CALL DUMP_FILTER(BFILT,NX,NY,14,FNAME2)
    CSCAN=TEMP
END IF

RETURN
END

```

\$cds on

SUNROUTINE BLOWUP (DATA,NX,NY,NXP,NYP,ILP,JLP,NXSECT,NYSECT,ALAM,DSA)

C Program last revised: 20 OCT 1999

C Replaces sector of data at beginning of array, transform to
C space domain and zero-files to increase resolution, then
C transforms back to wave-number domain.

EMA DATA
COMPLEX DATA(NX*NY)
INTEGER*4 K,KP

C NX,NY Dimensions of orginal array
C NXP,NYP Dimensions of sector before resolution enhancement
C NXSECT,NYSECT Dimensions of sector after enhancement
C ILP,JLP Starting indices of sector within orginal array
C ALAM Wavelength
C DSA Area factor for FFT from spectrum to aperture
C (In general, DSA = DSX * DSY = Kx/Ko * Ky/Ko)
C DA Area factor for FFT from aperture to spectrum
C (In general, DA = DX * DY / 4*PI**2)

C Download sector:

DO J=1,NYP
 DO I=1,NXP
 KP = (J-1)*NXP + I
 K = (JLP+J-2)*NX + ILP+I-1
 DATA(KP) = DATA(K)
 END DO
END DO

C Zero-fill in space domain:

CALL FFT2 (-1,NXP,NYP,DSA,DATA)
C (New) DX = ALAM / (NXP*DSX)
C (New) DY = ALAM / (NYP*DSY)
CALL EXPAND (DATA,NXP,NYP,NXSECT,NYSECT)

C Return to wave-number domain:

DA = ALAM**2 / (NXP*NYP*DSA) / (4. * PI**2)
CALL FFT2 (1,NXSECT,NYSECT,DA,DATA)
C (New) DSX = ALAM / (NXSECT*DX)
C (New) DSY = ALAM / (NYSECT*DY)

RETURN
END

\$CDS ON

SUBROUTINE CONVERT (DATA,NX,NY)

C Program last revised: 20 OCT 1999

C Converts the complex array DATA passed in rectangular form to
C polar form, with phase in degrees and the amplitude in dB
C with a floor of -200dB.

COMPLEX DATA(NX,NY)
EMA DATA

PI = ACOS(-1.)
RD = 180 / PI

DO J=1,NY

DO I=1,NX

X = REAL(DATA(I,J))
Y = AIMAG(DATA(I,J))

PHASE = ATAN2(Y,X) * RD !PHASE IN DEGREES

AMP = SQRT(X**2+Y**2)

IF (AMP .LE. 1.E-10) THEN
AMP=-200.

ELSE

AMP=20*ALOG10(AMP) !AMP IN dB

END IF

DATA(I,J) = CMPLX (AMP,PHSAE)

END DO

END DO

RETURN
END

\$CDS ON

SUBROUTINE CORREC(R01X1,R01Y1,R01X2,R01Y1,S10X,S10Y,D1,D2)

C Program last revised: 20 OCT 1999

C Performs probe correction for two polarization measurement in
C X,Y coordinates.

COMPLEX R01X1,R01X2,R01Y1,R01Y2,S10X,S10Y,D1,D2,DEL
DEL = R01X1 * R01Y2 - R01Y1 * R01X2
S10X = (D1 * R01Y2 - D2 * R01Y1) / DEL
S10Y = (D2 * R01X1 - D1 * R01X2) / DEL
RETURN
END

```

$CDS ON
!-----!
!
! SUBROUTINE DTAETIME          Program last revised: 20 OCT 1999 !
!
!      This routine gets the current date and time from the system !
!      clock and returns them in two integer arrays as follows:    !
!
!      IDTAE(1) = 1-digit year code
!      IDTAE(2) = month code (1-12)
!      IDTAE(3) = day (1-31)
!      ITIME(1) = hours (0-23)
!      ITIME(2) = minutes
!      ITIME(3) = seconds
!
! Subroutines called:
!None
!-----!

SUBROUTINE DTAETIME (IDTAE, ITIME)

INTEGER IDATE(3), ITIME(3), ITIME11(5), IYEAR, IBUFF(15)
CHARACTER FBUFF*30, MONTH*4
EQUIVALENCE (FBUFF,IBUFF)

CALL EXEC (11,ITIME11,IYEAR)          !Numerical time
CALL FTIME (IBUFF)                    !Formatted time

IDATE(1) = IYAER - 1900
ITIME(1) = ITIME11(4)
ITIME(2) = ITIME11(3)
ITIME(3) = ITIME11(2)

READ (FBUFF.90) IDATE(3), MONTH

90 FORMAT (16X, I1, 2X, A4)

IF (MONTH .EQ. 'JAN.') IDATE(2) = 1
IF (MONTH .EQ. 'FEB.') IDATE(2) = 2
IF (MONTH .EQ. 'MAR.') IDATE(2) = 3
IF (MONTH .EQ. 'APR.') IDATE(2) = 4
IF (MONTH .EQ. 'MAY.') IDATE(2) = 5
IF (MONTH .EQ. 'JUNE') IDATE(2) = 6
IF (MONTH .EQ. 'JULY') IDATE(2) = 7
IF (MONTH .EQ. 'AUG.') IDATE(2) = 8
IF (MONTH .EQ. 'SEPT') IDATE(2) = 9
IF (MONTH .EQ. 'OCT.') IDATE(2) =10
IF (MONTH .EQ. 'NOV.') IDATE(2) =11
IF (MONTH .EQ. 'DEC.') IDATE(2) =12

RETURN
END

```

\$CDS ON

\$EMA /BUFFER/

```
SUBROUTINE DUMP_FILTER(BFILT,NX,NY,IUNIT,FNAME)
CHARACTER CAXIS*1,CSCAN*80,NAME*15,POL*8,CTEMP*1,C1*1,FNAME*15
COMPLEX BFILT(NX,NY)
EMA BFILT
COMMON /PARAM/RSCAN(7),CAXIS,POL,CSCAN,NAME,DTAE(3),ITIME(3)
COMMON /BUFFER/ABUF(4096),PBUF(4096),IBUF
COMMON /MINMAX/AMIN,AMAX,PMIN,PMAX,MAXROW,MAXCOL
COMMON /USER/IWRITE,IREAD
```

C

SUBROUTINE TO DUMP FILTER OUT TO FILE

```
CTEMP=CAXIS
CAXIS='R'           !STORE BY ROWS
NAME=FNAME

AMIN=100.
AMAX=- 100.
PMIN=180.           !INITIAL VALUES
PMAX=- 180.

CALL DATETIME(IDATE,ITIME)

DO I=1,NY
    DO J=1,NX
        ABUF(J)=REAL(BFILT(J,I))           !AMPLITUDE
        PBUF(J)=AIMAG(BFILT(J,I))          !PHASE
    END DO
    IROW=I
    CALL WRITE_DATA(IUNIT,IROW,2,2)
END DO

CALL HEADWRITE(IUNIT,2)
CLOSE(UNIT=IUNIT,IOSTAT=IERR)
IF (IERR .GT. 0) THEN
    WRITE(1,*) 'ERROR ON CLOSING FILE'
END IF

RETURN
END
```

\$CDS ON

SUBROUTINE EEU(U,ETE)

C Program last revised: 20 OCT 1999

C Theoretical probe pattern in E-plane (F2 in memo).

COMMON /MVGE/ A,B,K

REAL K

COMPLEX ETE,ARGC

IF (U*U .GT. 1) THEN

ETE = (0.,0.)

ELSE

ARG = K * U * B / 2

ETE = SQRT(SINX(ARG))

ARGC = CSQRT(CMPLX(1.0 - U * U,0.0))

ARGC = -K * B * 0.25 * (1.0 - ARGC)

ETE = ETE * CEXP(ARGC)

END IF

RETURN

END

\$CDS ON

SUBROUTINE EHU(U,ETH)

C Program last revised: 20 OCT 1999

C Theoretical probe pattern in H-plane(F1 in memo)

COMMON /MVGE/ A,B,K

REAL K

COMPLEX ETH

PI = 3.141592654

IF (U*U .GE. 1.) THEN

ETH = (0.,0.)

ELSE

ARG1 = K * U * A / 2.

ARG2 = K * U * A / PI

ARG2 = 1.0 - ARG2 * ARG2

IF (ABS(ARG2) .LE. .0001) THEN

ETH = PI / 4.

ELSE

ETH = COS(ARG1) / ARG2

END IF

END IF

RETURN

END

\$CDS ON

SUBROUTINE EXPAND(DATA,MX,MT,NX,NY)

C Program last revised: 20 OCT 1999

C EXPAND moves the data array (DATA(MX,MY)) into the
C center of a larger array (DATA(NX,NY)) and zeros the extra
C elements(0.,0.).

EMA DATA
COMPLEX DATA(NX,NY),TEMP
INTEGER*4 K,II,JJ,IJ,IO,JO

MX1 = (NX-MX+1)/2
MY1 = (NY-MY+1)/2
MX2 = MX1 + MX
MY2 = MY1 + MY

DO J=NY,1,-1
JO = J-MY1 !L COORD. IN OLD ARRAY
DO I=NX,1,-1
IF (J.LE.MY1 .OR. J.GT.MY2) THEN
DATA(IJ) = (0.,0.)
ELSE IF (I.LE.MX1 .OR. I.GT.MX2) THEN
DATA(IJ) = (0.,0.)
ELSE
IO = I - MX1 !I COORD. IN OLD ARRAY
K = (JO-1)*MX + 1 !ABSOLUTE (1-DIM.) POSITION
JJ = (K-1)/NX + 1 !OLD ELEMENT POSITION IN
II = K - (JJ-1)*NX !NEW ARRAY
DATA(IJ) = DATA(II,JJ)
END IF
END DO
END DO

RETURN
END

\$CDS ON

SUBROUTINE FFT2 (ISN ,NX, NY, DA, DATA)

C Program last revised: 20 OCT 1999

C Routine to calculate the fast Fourier Transform or the
C inverse FFT of an input two-dimensional, complex array
C (DATA). Returns result in the same array.

C
C NX and NY are the dimensions of the array DATA and must
C be non-negative integer powers of 2.

C
C ISN is the control variable equal to +1 or -1.
C (ISN is the sign of the exponent.)

C
C DA is an area correction factor.

C
C The original of both input and output coordinates system are
C located at the (NX/2+1,NY/2+1) point of the array.

C

EMA DATA
COMPLEX DATA(NX,NY),T1,T2
REAL P11,SO,CO,S1,C1,SN,CS,SOISN
COMMON /USER/ IWRITE,IREAD

C IF(IABS(ISN).NE.1)GO TO 24

C WRITE(1,*) 'DA=',DA

PI2=2.ACOS(-1.)

IX=-1

M=0

DO WHILE (NX .GT. M)

IX=IX+1

M=2**IX

IF (NX .LT. M) THEN

WRITE (IWRITE,*) 'FFT ERROR: NX must be a power of 2.'

STOP

END IF

END DO

IY=-1

M=0

DO WHILE (NY .GT. M)

IY=IY+1

M=2**IY

IF (NY .LT. M) THEN

WRITE (IWRITE,*) ' FFT ERROT: NY must be a power of 2.'

STOP

END IF

END DO

NX2=NX/2

NY2=NY/2

DO I=1,NX2,1

I1=I+NX2

DO J=1,NY,1

T1=DATA(IJ)

DATA(IJ)=DATA(I1,J)

DATA(I1,J)=T1

END DO

END DO

DO J=1,NY2,1

J1=J+NY2

DO I=1,NX,1

T2=DATA(IJ)

DATA(IJ)=DATA(I,J1)

DATA(I,J1)=T2

```

        END DO
END DO

NXBIT=16- IX
NX1=NX- 2
DO I=1,NX1,1
    IFLIP=0
    DO J=NXBIT,15,1
        N=NXBIT- J
        N=N+15
        IFLIP=2*IFLIP+IAND(ISHFT C(I,N+1,16),1)
    END DO
    IF (I.GT.IFLIP) THEN
        I1=I+1
        I2=IFLIP+1
        DO J=1,NY,1
            T1=DATA(I2,J)
            DATA(I2,J)=DATA(I1,J)
            DATA(I1,J)=T1
        END DO
    END IF
END DO

NYBIT=16- IY
NY1=NY- 2
DO J=1,NY1,1
    JFLIP=0
    DO I=NYBIT,15,1
        M=NYBIT- 1
        M=M+15
        JFLIP=2*JFLIP+IAND(ISHFT C(J,M+1,16),1)
    END DO
    IF(J.GT.JFLIP) THEN
        J1=J+1
        J2=JFLIP+1
        DO I=1,NX,1
            T2=DATA(I,J2)
            DATA(I,J2)=DATA(I,J1)
            DATA(I,J1)=T2
        END DO
    END IF
END DO

DO I=1,IX,1
    NEL=2**I
    NEL2=NEL/2
    NSET=NX/NEL
    SI=SIN(PI2/NEL)
    CI=COS(PI2/NEL)
    DO K=1,NSET,1
        INCR=(K- 1)*NEL
        SO=0.0
        CO=1.0
        DO L=1,NEL2,1
            I1=L+INCR
            I2=I1+NEL2]
            DO J=1,NY,1
                T1=DATA(I1,J)
                SOISN=SO*(FLOAT(ISN))
                T2=DATA(I2,J)*CMPLX(CO,SOISN)
                DATA(I1,J)=T1+T2
                DATA(I2,J)=T1- T2
            END DO
            SN=SO*CI+CO*SI
            CS=CO*CI- SO*SI
            CO=CS
            SO=SN
        END DO
    END DO
END DO
END DO

```



```

DO I=1,NX2,1
    I1=I+NX2
    DO J=1,NY,1
        T1=DATA(I,J)
        DATA(I,J)=DATA(I1,J)
        DATA(I1,J)=T1
    END DO
END DO

DO J=1,NY2,1
    J1=J+NY2
    DO I=1,NX,1
        T2=DATA(I,J)
        DATA(I,J)=DATA(I,J1)
        DATA(I,J1)=T2
    END DO
END DO

IF (DA .NE. 1.) THEN
    DO J=1,NY
        DO I=1,NX
            DATA(I,J)=DATA(I,J)*DA
        END DO
    END DO
END IF

RETURN

C 24 CONTINUE
C      WRITE(6,*0 ' ISN IS NOT +1 OR -1 IN FFT2'
C      RETURN

END

```

```

$CDS ON
      REAL FUNCTION G0WAVEGD ()

      COMMON /WVGE/ A,B,AK0

C      INITIALIZATION

      EPS = .001
      STANDARD = .01
      PI = ACOS(-1.)
      ALAM = 2. * PI / AK0
      BETA0 = SQRT(1. - (PI / (AK0 * A))**2)
      G01=0.
      N = 62

C      REPEAT

10 N = N * 2
      DELTHETA = PI / (N - 1)
      G02 = G01
      AMAXTHE = 7 * PI / 12
      D = 0.
      I = 0

20 I = I + 1
      THETA = (I - 1) * DELTHETA
      IF (THETA .LE. AMAXTHE) THEN
          DD1 = (1. + BETA0) * AK0 * B / 2. * SIN(THETA)
          DN1A = 1. + BETA0 * COS(THETA)
          DN1B = SIN(AK0 * B / 2. * SIN(THETA))
          DN1 = DN1A * DN1B
          D1 = (DN1 / DD1)**2
          IF (ABS(SIN(THETA) - (ALAM / (2. * A))) .LT. EPS) THEN
              D2 = (PI * COS(THETA) / 2)**2
          ELSE
              DD2 = (PI / 2.)**2 - (AK0 * A / 2 * SIN(THETA))**2
              DN2 = COS(THETA) * COS(AK0 * A / 2. * SIN(THETA))
              D2 = ((PI / 2.)**2 * (DN2 / DD2))**2
          END IF
      END IF

30 D = (D1 + D2) * SIN(THETA) * DELTHETA + D
      IF ((THETA + DELTHETA) .LT. (PI - EPS)) GOTO 20
      G01 = 4 / D
      DIFF = ABS(G02 - G01) / G01
      IF ((DIFF .GT. STANDARD) .AND. (N .LE. 1000)) GOTO 10
      IF (N .GT. 1000) THEN
          WRITE(1,*) 'WARNING: Probe gain fails to converge.'
      END IF
      G0WAVGD = G01

      RETURN

      END

```

\$cds on

```
SUBROUTINE GETPAT (J,NY,AMP1X,PHASE1Y,AMP2X,PHASE2Y)

DIMENSION AMP1X(4096), PHASE1Y(4096), AMP2X(4096), PHASE2Y(4096)

EMA AMP1X,AMP2X,PHASE1Y,PHASE2Y

DTOR = ACOS(-1.)/180.                ! degrees to radians

CALL READ_DATA (8, J, 2, 2, AMP1, PHASE1, JUNK)
CALL READ_DATA (9, J, 2, 2, AMP2, PHASE2, JUNK)

DO J=1, NY
    AMP = 10.**(AMP1X(J)/20.)
    PHASE = PHASE1Y(J)*DTOR
    AMP1X(J) = AMP *COS(PHASE)

    AMP = 10.**(AMP2X(J)/20.)
    PHASE = PHASE2Y(J)*DTOR
    AMP2X(J) = AMP*COS(PHASE)
    PHASE2Y(J) = AMP*SIN(PHASE)
END DO

RETURN

END
```

\$CDS ON

```
SUBROUTINE GRIDSET(MAXPTS, ITIT,ISTARTX,ISTARTY,MX,MY,NX,NY,
+
IXINC, IYINC)
CHARACTER CAXIS*1,POL*8, CSCAN*80,NAME*15,CSTEP*10,TEMP*80
COMMON /PARAM/RSCAN(7),CAXIS,POL,CSCAN,NAME,IDATE(3),ITIME(3)
COMMON /USER/IWRITE,IREAD
```

C Program last revised: 20 OCT 1999

```
C SUBROUTINE TO PROMPT USER FOR OPTION TO DETERMINE GRID OF DATA
C TO BE USED FOR PLOTTING OR LISTING.
C MAXPTS---SUPPLIED BY CALLING ROUTINE. DETERMINES MAXIMUM NO. OF
C PTS TO BE PLOTTED OR LISTED
C ITIT---SUPPKIED BY CALLING ROUTINE. DETERMINE IF USER IS PROMPTED
C FOR TITLE
C ALL FOLLOWING VALUES ARE RETURNED BY GRIDSET
C ISTARTX---STARTING X PT TO BE PLOTTED
C ISTARTY---STARTING Y PT TO BE PLOTTED
C MX---THE NUMBER OF X PTS TO BE PLOTTED
C MY---THE NUMBER OF Y PTS TO BE PLOTTED
C NX---THE LAST X PT TO BE PLOTTED
C NY---THE LAST Y PT TO BE PLOTTED
C IXINC---THE X THINNING INCREMENT
C IYINC---THE Y THINNING INVREMENT
```

```
NX=RSCAN(3)
NY=RSCAN(6)
```

```
WRITE(IWRITE,*) 'ENTER CARR. REP. TO DEFAULT THE FOLLOWING QUESTIONS'
WRITE(IWRITE,*)
```

```
IF (ITIT .EQ. 1) THEN
    WRITE(IWRITE,*) 'THR CURRENT TITLE ID:'
    WRITE(IWRITE,*) CSCAN
    WRITE(IWRITE,*) 'ENTER THE TITLE YOU WOULD LIKE TO PRINT'
    READ(IREAD,10) temp
    IF (TEMP .GT. ' ') CSCAN=TEMP
END IF
```

```
3 WRITE(IWRITE,*) 'ENTER X AXIS STARTING, ENDING PT. TO BE PLOTTED(1,'NX,')'
READ(IREAD, 10) CSTEP
IF (CSTEP .GT. ' ') THEN
    READ(CSTEP,*) ISTARTX,IENDX
    IF (ISTARTX .LT. 1 .OR. ISTARTX .GT. IENDX) GOTO 3
    IF (IENDX .GT. NX) GOTO 3
ELSE
    ISTARX=1
    IENDX=NX
END IF
```

```
4 WRITE(IWRITE,*) 'ENTER Y AXIS STARTING,ENDING PT. TO BE PLOTTED(1,'NY,')'
READ(IREAD,10)
IF (CSTEP .GT. ' ') THEN
    READ(CSTEP,*) ISTARTY,IENDY
    IF (ISTARTY .LT. 1 .OR. ISTARTY .GT. IENDY) GOTO 4
    IF (IENDY .GT. NY) GOTO 4
ELSE
    ISTARTY=1
    IENDY=NY
END IF
```

```
XSTEP=(IENDX-ISTARTX+1)/FLOAT(MAXPTS)
YSTEP=(IENDY-ISTARTY+1)/FLOAT(MAXPTS)
```

```
IF (XSTEP .LE. 1) THEN
    IXINC=1
ELSE IF (XSTEP .NE. INT(XSTEP)) THEN
    IXINC=INT(XSTEP+1.)
ELSE
    IXINC=INT(XSTEP)
END IF
```

```

        IF (YSTEP .LE. 1) THEN
            IYINC=1
        ELSE IF (YSTEP .NE. INT(YSTEP)) THEN
            IYINC=INT(YSTEP+1.)
        ELSE
            IYINC=INT(YSTEP)
        END IF

16 WRITE(IWRITE,*)'ENTER X AXIS THINNING INCREMENT(INTEGER .GE. ' ,IXINC,')'
   READ(IREAD,10) CSTEP
      IF (CSTEP .GT. ' ') THEN
         READ(CSTEP,*) IX
         IF (IX .LT. IXINC) GOTO 16
         IXINC=IX
      END IF

18 WRITE(IWRITE,*)'ENTER Y AXIS THINNING INCREMENT(INTEGER .GE. ' ,IYINC,')'
   READ(IREAD,10) CSTEP
      IF (CSTEP .GT. ' ') THEN
         READ(CSTEP,*) IY
         IF (IY .LT. IYINC) GOTO 18
         IYINC=IY
      END IF

      MX=1 + (IENDX- ISTARTX)/IXINC           !# OF X PTS
      MY=1 + (IENDY- ISTARTY)/IYINC           !# OF Y PTS

      NX=ISTARTX+(MX-1)*IXINC                  !LAST X PT
      NY=ISTARTY+(MY-1)*IYINC                  !LAST Y PT
10 FORMAT(A)
      RETURN
      END

```

\$CDS ON

```
!-----!  
! SUBROUTINE HEADER      Program last revised: 20 OCT 1999  !  
!                                     !  
! Entry points:                                     !  
! HEADREAD                                     !  
!                                     !  
! HEADWRITE                                     !  
! This routine reads or writes the header record of a data !  
! file depending on which entry point is used.             !  
! IUNIT - Unit number of the data file.                   !  
! IRDAT - Indicates whether amplitude and/or               !  
! phase information is stored in the file.                 !  
!                                     !  
! Subroutine called:                                     !  
! None                                                     !  
!-----!
```

SUBROUTINE HEADER

```
COMMON /PARAM/ RSCAN(7), CAXIS, POL, CSCAN, NAME, IDATE(3), ITIME(3), NPOL  
COMMON /MINMAX/ AMIN, AMAX, PMIN, PMAX, MAXY, MAXX  
COMMON /USER/ IWRITE, IREAD
```

```
CHARACTER CAXIS*1, POL*8, CSCAN*80, NAME*15
```

```
ENTRY HEADWRITE (IUNIT, IRDAT)      ! TO WRITE THE HEADER RECORD
```

```
INQUIRE(UNIT=IUNIT, IOSTAT=IERR, ERR=17, RECL=IRECLB)  !GET RECORD LENGTH  
NDUM=(IRECLB-168)/2      !NUMBER OF DUMMY VAR. TO WRITE OUT  
WRITE(UNIT=IUNIT, IOSTAT=IERR, ERR=17, REC=1)  
RSCAN, CAXIS, POL, CSCAN, NAME, IDATE, ITIME, AMIN, AMAX, PMIN, PMAX, MAXY, MAXX, IRDAT,  
+ NPOL, (IDUM, I=1, NDUM)
```

```
17 IF (IERR .GT. 0) THEN  
    WRITE(IWRITE,*) 'ERROR', IERR, 'WRITING HEADER'  
    PAUSE
```

```
END IF
```

```
RETURN
```

C

```
ENTRY HEADREAD(IUNIT, IRDAT)      ! TO READ THE HEADER RECORD
```

```
READ (UNIT=IUNIT, IOSTAT=IERR, ERR=27, REC=1) RSCAN, CAXIS, POL, CSCAN,  
+ NAME, IDATE, ITIME, AMIN, AMAX, PMIN, PMAX, MAXY, MAXX, IRDAT, NPOL
```

```
27 IF (IERR .GT. 0) THEN  
    WRITE(IWRITE,*) 'ERROR', IERR, 'READING HEADER'
```

```
END IF
```

```
RETURN
```

```
END
```

\$CDS ON

```
!-----!
!
! SUBROUTINE NAMFILE      Program last revised: 20 OCT 1999      !
!
!      This subroutine opens a datafile for subsequent reads or !
!      writes. IUNIT is the unit number to be associated with !
!      the file. ISTATUS is the status of the file:             !
!      ISTATUS = 0 - New file                                     !
!      = 1 - Old file                                           !
!      = 2 - Status unknown                                     !
!      DDIR is the data directory, if other than                !
!      ::XYZFILES                                               !
!
!      LGBUF is a library subroutine to enlarge I/O buffer size. !
!      NOTE: the buffer array LBUF must not be in EMA under any !
!      circumstance.                                           !
!      NOTE: if CDS is used, then either the common block      !
!      /RECBUFF/must be declared in the main program and       !
!      this subroutine, or the call to LGBUF must be made      !
!      in the main program (in which case /RECBUFF/ is not     !
!      required.)                                              !
!
!      Subroutine called:                                       !
!      DATETIME                                                !
!-----!
```

```
C      SUBROUTINE NAMFILE (IUNIT, ISTATUS,DDIR)
      SUBROUTINE NAMFILE (IUNIT, ISTATUS)
```

```
      COMMON /PARAM/ RSCAN(7), CAXIS, POL, CSCAN, NAME,IDATE(3), ITIME(3), NPOL
      COMMON /RECBUFF/ LBUF(8200)
      COMMON /ISER/ IWIRTE, IREAD
```

```
      CHARACTER CAXIS*1, POL*8,CSCAN*80, NAME*15
      CHARACTER DDIR*16, INFILE*30, STAT*7
```

```
C      NP = PCOUNT()                                ! Number of parameters passed
C      IF (NP .LT. 3) DDIR = '/XYZFILES '
```

```
C      ID = INDEX (DDIR, ' ') - 1                    !Length of string
C      IF (ID .LE. 0) ID=16
```

```
5      WRITE (IWRITE,*) 'Enter data file name:'
      READ (IREAD,20) NAME
20     FORMAT(A)
C      INFILE = DDIR(1:ID)// '/' // NAME
      INFILE = NAME// '/' ::XYZFILES'
```

```
      IF (ISTATUS .EQ. 0) STAT='OLD '
      IF (ISTATUS .EQ. 1) STAT='NEW '
      IF (ISTATUS .EQ. 2) STAT='UNKNOWN'
```

```
      IF (STAT .EQ. 'NEW') THEN
          NPTS=RSCAN(6)
          IF (CAXIS .EQ. 'X') NPTS=RSCAN(3)
          IRECLB=(NPTS*4)+2 !RECORD LENGTH(BYTES)- -- AMP OR PHASE AND STATUS
          IF (IRECLB .LT. 180) IRECLB=180 !INSURE ENOUGH ROOM FOR HEADER REC.
          CALL DTAETIME(IDATE, ITIME)
      ELSE
          INQUIRE(FILE=INFILE,IOSTAT=IERR,ERR=65,RECL=IRECLB) !READ RECORD LTH
      END IF
```

```
                                OPEN(UNIT=IUNIT,FILE=INFILE,ACCESS='DIRECT',FORM=UNFORMATTED',
RECL=IRECLB,IOSTAT=IERR,ERR=65,STATUS=STAT)
```

```
65     IF (IERR .GT. 0) THEN
          WRITE(IWRITE,*) 'ERROR ', IERR, ' ON OPENING FILE'
          GOTO 5
```

```
ELSE
CALL LGBUF (LBUF,IRECLB/2)      !ENLARGE I/O BUFFER TO #BYTES/2
END IF

RETURN

END
```



```
SUBROUTINE NFNORM (DATA,NX,NY,AKX,AKY)
```

```
CHARACTER CSCAN*80,CAXIS*1,POL*8,NAME*15  
COMMON /PARAM/ RSCAN(7),CAXIS,POL,CSCAN,NAME,IDATE(3),ITIME(3)  
COMMON /MINMAX/ AMIN,AMAX,PMIN,PMAX,MAXY,MAXX  
COMPLEX DATA(NX,NY), CJ
```

```
CJ = (0.,1.)  
DR = ACOS(- 1.) / 180.  
XINC = RSCAN(2)  
YINC = RSCAN(5)  
MX = NA/2 + 1  
MY = NY/2 + 1
```

```
DO J=1,NY  
    PY = (J-MY)*YINC*AKY  
    DO I=1,NX  
        PX = (I-MX)*XINC*AKX  
        AMP=10.*( (REAL(DAT A(I,J))- AMAX) /20.)  
        PHS= (AIMAG(DAT A(I,J))- PMAX)*DR  
        DAT A(I,J)=AMP*CEXP(CJ* (PHS + (PX+PY) ))  
    END DO  
END DO  
  
RETURN  
  
END
```

```

$CDS ON
FUNCTION PCALC (GAM, SX, SY, S10X, S10Y)

C      Program last revised: 20 OCT 1999

C      Incremental calculation used to accumulate total power sum.

COMPLEX S10X,S10Y,B1,B2

PCALC = 0.

IF (GAM.EQ.0) THEN
    PCALC = CABS(S10X)**2+CABS(S10Y)**2
ELSE IF (GAM .LT. 0.9999) THEN
    SZ=SQRT(1- GAM)

C      B1,B2 are b-sub-q(m,k), scalar spectral density functions(P.55)
C      Kerns 1.2-1.5a, p.57
    B1 = (SX*S10X+SY*S10Y)
    B2 = (-SY*S10X+SX*S10Y)

C      Kerns 1.4-9, p. 65
    PCALC = CABS(B1)**2/SZ + CABS(B2)**2*SZ
END IF

RETURN

END

```

\$CDS ON

SUBROUTINE PCORR (DATA,NX,NY,DATA2,NX2,NY2,ICORR,IPRBR,NPOL,NPOUT,POLIN,POLOUT)

CC

C
C Subroutine to do probe correction, accumulate power sum, and
C convert to desired output polarization.
C
C Program last revised: 20 OCT 1999
C
C
CC

COMMON /USER/ IWRITE,IREAD
COMMON /WVGE/ A,B,AK0
COMMON /PARAM/ RSCAN(7),CSXIS,POL,CSCAN,NAME,IDATE(3),ITIME(3)
COMPLEX D1,D2
COMPLEX DATA(NX,NY),DATA2(NX2,NY2),F1A,F1B,F2A,F2B,FA,FB
COMPLEX SD1X,SD1Y,SD2X,SD2Y,SD1XR,SD1YR,SD2XR,SD2YR,S10X,S10Y
DIMENSION PPAT1X(4096),PPAT1Y(4096),PPAT2X(4096),PPAT2Y(4096)
CHARACTER CAXIS*1,POL*8,CSCAN*80,NAME*15
EMA DATA,DAT A2,PPAT1X,PPAT1Y,PPAT2X,PPAT2Y

C Z0 is characteristic impedance of transmission line to probe.

Z0 = 50 !50Ohm
PI = ACOS(-1.)
LAMA = 2. * PI / AK0
SX0 = RSCAN(1)
SY0 = RSCAN(4)
SXINC = RSCAN(2)
SYINC = RSCAN(5)
POWER = 0.
CPOLI = COS(POLIN)
SPOLI = SIN(POLIN)
CPOLO = COS(POLOUT)
SPOLO = SIN(POLOUT)

C GMAX is the probe gain on axis.

C ALAMGP2 = 1. / (1 - (ALAM / (2 * A))**2)
C IF (ALAMGP2 .GT. 0) THEN
C ZPRIME = SQRT(ALAMGP2)
C PTRANS = 4. * ZPRIME / (1 + ZPRIME)**2
C ELSE
C ZPRIME = 1.
C WRITE(IWRITE,*) 'WARNING: Probe dimensions too small',
C 'in subroutine PCORR.'
C PTRANS = 1.
C END IF
C GMAX = PTRANS *32 * A * B / (PI *8 ALAM**2)
C IF (ICORR.EQ.0) THEN
C GMAX=G0WAVGD()
C SMAX =.0164*ALAM*SQRT(GMAX)
C ELSE
C SMAX=1
C END IF

C SMAX is the probe spectrum peak as defined by Kerns 1.6-19
C and 1.6-21a, page 76-77.

C SMAX = SQRT(GMAX)*(4*PI*AK0**2*377/Z0)**-.5

C (Where Z0 is transmission line impedance to the probe -50 ohm)

C For gain relative to available power, use the factor
C SQRT (4 * PI * Z0*AK0**2 / 377)
C (See Kerns, 1.6-6, p.74)

GAINFAC=SQRT(4*PI*AK0**2*Z0/377)

IF (ICORR .GE. 0) THEN

```

C          Probe correction (Polarizations are A and B):

DO J=,NY
    SY=SY0+(J-1)*SYINC
    IF (ICORR.GT.0) CALL GETPAT (J, NY, PPAT1X, PPAT1Y,
                                PPAT2X,PPAT2Y)
                                +

DO I=1,NX
    SX=SX0+(I-1)*SXINC

    D1 = DATA(I,J)
    IF (NPOL.EQ.2) THEN
        D2=DATA2(I,J)
    ELSE
        D2=(0.,0.)
    END IF

    GAM = SX*SX + SY*SY
    IF (GAM.GE..9999) THEN
        D1 = (0.,0.)
        D2 = (0.,0.)
    ELSE
        UA = -CPOLI*SX + SPOLI*SY !Aperture position relative
        VA = SPOLI*SX + CPOLI*SY !    to probe orientation
        UB = -VA * IPRBR          ! Ditto, after probe

        VB = UA * IPRBR          !

        IF (ICORR.EQ.0) THEN
            CALL EHU(UA,F1A)      !Theoretical probe
            CALL EEU(VA,F2A)      ! for principal planes
            CALL EHU(UB,F1B)      !
            CALL EEU(VB,F2B)      !

rotation

pattern

C          An electric source spectrum is assumed. Huygens must be converted
C          before using

FA = F1A*F2A*SMAx
FB = F1B*F2B*SMAx

SD1X = FA*SPOLI
SD1Y = FA*CPOLI
SD2X = -FB*CPOLI*IPRBR
SD2Y = FB*SPOLI*IPRBR
ELSE
    SD1X = PPAT1X (I)
    SD1Y = PPAT1Y (I)
    SD2X = PPAT2X (I)
    SD2Y = PPAT2Y (I)
END IF

C          Convert transmit probe spectra to receive spectra :
CALL S10T01(SD1X,SD1Y,SX,SY,SD1XR,SD1YR)
CALL S10T01(SD2X,SD2Y,SX,SY,SD2XR,SD2YR)

C          Probe correction :
CALL CORREC(SD1XR,SD1YR,SD2XR,SD2YR,S10X,S10Y,D1,D2)
D1 = S10Y
D2 = S10X

C          Accumlate total power sum :
IF (ICORR.EQ.0) POWER = POWER + PCALC(GAM,SX,SY,S10X,S10Y)

C          Convert to output polarization :
C
IF (NPOUT.EQ.1) CALL XYTYCON(SX,SY,S10X,S10Y,SPOLO,CPOLO,D1,D2)
IF (NPOUT.EQ.2) CALL XYTHUY(SX,SY,S10X,S10Y,SPOLO,CPOLO,D1,D2)

```

```

                                IF (NPOUT.EQ.3) CALL XYTZCON(SX,SY,S10X,S10Y,SPOLO,CPOLO,D1,D2)
                                IF (NPOUT.NE.0) THEN
                                    D1 = D1 * GAINFAC
                                    D2 = D2 * GAINFAC
                                END IF
                            END IF

                            DATA(IJ) = D1
                            IF (NPOL.NE.1) DATA2(IJ) = D2
                        END DO
                    END DO

                ELSE

C                    No probe correction :

                    DO J=1,NY
                        SY=SY0+(J-1)*SYINC
                        DO I=1,NX
                            SX=SX0+(I-1)*SXINC

                            D1 = DATA(I,J)/SMAX
                            IF (NPOL.EQ.2) THEN
                                D2 = DATA2(I,J)/SMAX
                            ELSE
                                D2 = (0.,0.)
                            END IF

                            GAM = SX*SX + SY*SY
                            IF (GAM.GE.,9999) THEN
                                D1 = (0.,0.)
                                D2 = (0.,0.)
                            ELSE
C                                Notice that D1 is Y- component if no rotation
                                S10X =(- D2*CPOLI*IPRBR + D1*SPOLI)
                                S10Y =(D2*SPOLI*IPRBR + D1*CPOLI)

                                D1=S10Y
                                D2=S10X

                                IF (NPOUT.EQ.1) CALL XYTYCON(SX,SY,S10X,S10Y,SPOLO,CPOLO,D1,D2)
                                IF (NPOUT.EQ.2) CALL XYTHUY(SX,SY,S10X,S10Y,SPOLO,CPOLO,D1,D2)
                                IF (NPOUT.EQ.3) CALL XYTZCON(SX,SY,S10X,S10Y,SPOLO,CPOLO,D1,D2)
                                IF (NPOUT.NE.0) THEN
                                    D1 = D1 * GAINFAC
                                    D2 = D2 * GAINFAC
                                END IF

                                POWER = POWER + PCALC(GAM,SX,SY,S10X,S10Y)
                                END IF

                                DATA(IJ) = D1
                                IF (NPOL.NE.1) DATA2(IJ) = D2
                            END DO
                        END DO
                    END IF

C                    DELK=AK0**2*SYINC*SXINC
C                    POWER=POWER*DELK/(240.*PI)
C                    POW=20.*ALOG10(POWER)
C                    WRITE(1,*) 'TOTAL RADIATED POWER ID ',POW

                RETURN
            END

```

\$CDS ON

SUBROUTINE POLAR (DATA,AMP,PHA)

C

Program last revised: 20 OCT 1999

COMPLEX DATA

X = REAL(DATA)

Y = AIMAG(DATA)

AMP = SQRT(X**2+Y**2)

PHA = ATAN2(Y,X)

RETURN

END

\$CDS ON

SUBROUTINE POWER(N,NP2,NADD)

C

Program last revised: 20 OCT 1999

NP2=ALOG(FLOAT((N))/0.69314718+0.001

NP2=NP2+NADD

NP2=2**NP2

RETURN

END

\$CDS ON

```
!-----!  
!  
! SUBROUTINE READWRITE      Program last revised: 20 OCT 1999  !  
!  
! ENTRY POINTS            !  
! READ_DATA               !  
! WRITE_DATA              !  
!  
! Depending on which entry point is used, this routine reads  !  
! a row of data from, or writes a row of data to, a data file !  
!  
! IUNIT - Unit number of data file                             !  
! IROW - Number of the row or column to be transferred        !  
! IRDAT = 0 - only amplitude is recorded                      !  
!   = 1 - only phase is recorded                             !  
!   = 2 - amplitude and phase are recorded                   !  
! IDATA = 0 - only amplitude information is transferred       !  
!   = 1 - only phase information is transferred              !  
!   = 2 - both amplitude and phase are transferred           !  
!  
! Subroutine called:                                           !  
! None                                                         !  
!-----!
```

SUBROUTINE READWRITE

EMA ABUF(4096), PBUF(4096)

COMMON /PARAM/ RSCAN(7),CAXIS,POL,CSCAN,NAME,IDATE(3), ITIME(3), NPOL
COMMON /USER/ IWRITE,IREAD

CHARACTER CAXIS*1, POL*8, CSCAN*80, NAME*15

C

ENTRY READ_DATA (IUNIT, IROW, IRDAT, IDATA, ABUF, PBUF, IBUF)

```
IF (CAXIS .EQ. 'X' ) THENV                                !DATA COLLECTED ALONG X AXIS  
    NPTS=RSCAN(3)                                          !# X PTS  
ELSE                                                         !DATA COLLECTED ALONG Y AXIS  
    NPTS=RSCAN(6)                                          !# Y PTS  
END IF
```

C

Section for reading data from a file

```
IF (IRDAT .NE. 2) THEN                                     !ONLY AMP OR PHASE STORED  
    IF (IDATA .NE. IRDAT) WRITE(IWRITE,*) 'WARNING- ---','DATA REQUESTED WAS NOT RECORDED'  
    IREC=1+IROW                                           !RECORD #  
    IF (IDATA .EQ. 0) READ(UNIT=IUNIT,Iostat=IERR,ERR=99,REC=IREC) (ABUF(M),M=1,NPTS),IBUF  
    IF (IDATA .EQ. 1) READ(UNIT=IUNIT,Iostat=IERR,ERR=99,REC=IREC) (PBUF(M),M=1,NPTS),IBUF  
ELSE                                                         !AMPLITUDE AND PHASE STORED  
    IREC=2+2*(IROW-1) !RECORD #  
    IF (IDATA .NE. 1) READ(UNIT=IUNIT,Iostat=IERR,ERR=99,REC=IREC)(ABUF(M),M=1,NPTS),IBUF  
    IF (IDATA .NE. 0) READ(UNIT=IUNIT,Iostat=IERR,ERR=99,REC=IREC+1)(PBUF(M),M=1,NPTS),IBUF  
  
END IF  
  
RETURN
```

C

ENTRY WRITE_DATA (IUNIT,IROW,IRDAT,IDATA,ABUF,PBUF,IBUF,AMIN, AMAX, PMIN, PMAX, MAXY,MAXX)

```
IF (CAXIS .EQ. 'X') THEN                                !DATA COLLECTED ALONG X AXIS  
    NPTS=RSCAN(3)                                          !# X PTS  
ELSE                                                         !DATA COLLECTED ALONG Y AXIS  
    NPTS=RSCAN(6)                                          !# Y PTS  
END IF
```

C

Section to determine maximum and minimum amplitude and phase


```

IF (IROW .EQ. 1) THEN
    AMIN=100.
    AMAX=- 100.
    PMIN=180.          !INITIALIZE THE MAX AND MINS
    PMAX=- 180.
END IF

DO I=1,NPTS
    IF(ABUF(I) .GT. AMAX) THEN
        AMAX=ABUF(I)          !AMPLITUDE MAX
        IF (CAXIS .EQ. 'X') THEN
            MAXY=IROW
            MAXX=I              !SECTION TO DETERMINE
                                !MAX AND MINS
        ELSE
            MAXY=I
            MAXX=IROW
        END IF
    END IF
    IF (ABUF(I) .LT. AMIN) AMIN=ABUF(I)          !AMP MIN
    IF (PBUF(I) .GT. PMAX) PMAX=PBUF(I)          !PHASE MAX
    IF (PBUF(I) .LT. PMIN) PMIN=PBUF(I)          !PHASE MIN
END DO

C      Section for writing data to a file

IF (IRDAT .NE. 2) THEN          !ONLY AMP OR PHASE STORED
    IREC=1+IROW                  !RECORD #
    IF (IRDAT .EQ. 0) WRITE(UNIT=IUNIT,IOSTAT=IERR,ERR=98,REC=IREC) (ABUF(M),M=1,NPTS),IBUF
    IF (IRDAT .EQ. 1) WRITE(UNIT=IUNIT,IOSTAT=IERR,ERR=98,REC=IREC) (PBUF(M),M=1,NPTS),IBUF
ELSE
    IREC=2+2*(IROW-1)           !RECORD #
    IF (IDATA .NE. 1) WRITE(UNIT=IUNIT,IOSTAT=IERR,ERR=98,REC=IREC) (ABUF(M),M=1,NPTS),IBUF
    IF (IDATA .NE. 0) WRITE(UNIT=IUNIT,IOSTAT=IERR,ERR=98,REC=IREC) (PBUF(M),M=1,NPTS),IBUF
END IF

RETURN

C      Section for error messages

98      WRITE(IWRITE,*) 'ERROR', IERR,'WRITING ROW',IROW,'TO FILE',NAME

RETURN

99      WRITE (IWRITE,*) 'ERROR',IERR,'READING ROW',IROW,'FROM FILE',NAME
RETURN

END

```

\$CDS ON

SUBROUTINE S10T01(S10X,S10Y,UX,UY,S01X,S01Y)

C Program last revised: 20 OCT 1999

C Uses reciprocity to convert S10 to S01 in Cartesian coordinates
C for a direction Ux,Uy. Note that S01X and S01Y are at -K and
C S10X and S10Y are at K.

COMPLEX S10X,S10Y,S01X,S01Y,GAM,ET1,ET2,A(2,2)
REAL KTSQ

KTSQ = UX * UX + UY * UY
IF (KTSQ .EQ. 0.) THEN
 A(1,1) = (1.,0.)
 A(1,2) = (0.,0.)
 A(2,1) = (0.,0.)
 A(2,2) = (1.,0.)
ELSE
 GAM = CSQRT(CMPLX(1.0 - KTSQ,0.0))
 ET1 = 1.0 /GAM
 ET2 = GAM
 A(1,1) = (ET1 * UX*UX + ET2*UY*UY) / KTSQ
 A(1,2) = (ET1 - ET2) * UX * UY / KTSQ
 A(2,1) = (A(1,2)
 A(2,2) = (ET1 * UY * UY + ET2 * UX * UX) / KTSQ
END IF

S01X = A(1,1) *S10X + A(1,2) * S10Y
S01Y = A(2,1) * S10X +A(2,2) * S10Y

RETURN
END

\$CDS ON

```
SUBROUTINE SEPARATE(XINC,YINC,NPOL,NX,NY,DATA,DATA2,CAXIS)
COMPLEX DATA(NX,NY),DATA2(NX,NY),SDATA(4096)
CHARACTER CAXIS*1
EMA DATA,DATA2,SDATA

      PI=ACOS(-1.)

      IF (CAXIS.EQ.'R') THEN
        DA=XINC/2./PI
        DO J=1,NY
          CALL FFT2 (1,NX,1,DA,DATA(1,J))
          IF (NPOL.EQ.2) CALL FFT2 (1,NX,1,DA,DATA2(2,J))
        END DO
      ELSE
        DA=YINC/2./PI
        DO I=1,NX
          DO J=1,NY
            SDATA(J)=DATA(I,J)
          END DO
          CALL FFT2(1,1,NY,DA,SDATA)
          CALL FFT2(1,NY,1,DA,SDATA)
          DO J=1,NY
            DATA(I,J)=SDATA(J)
          END DO
        END DO
        IF (NPOL.EQ.2) THEN
          DO I=1,NX
            DO J=1,NY
              SDATA(J)=DATA2(I,J)
            END DO
            CALL FFT2 (1,1,NY,DA,SDATA)
            DO J=1,NY
              DATA2(I,J)=SDATA(J)
            END DO
          END DO
        END IF
      END IF

      RETURN
      END
```

\$CDS ON

```
SUBROUTINE SEPTRANS(XINC,YINC,NPOL,NX,NY,DATA,DATA2,CAXIS)

COMPLEX DATA(NX,NY),DATA2(NX,NY),SDATA(4096)
EMA DATA,DATA2,SDATA
CHARACTER CAXIS*1
COMMON /WVGE/A,B,AK0
COMMON /TRANS/TX,TY,TZ,FILTER,SXINC,SYINC

PI=ACOS(- 1.)

IF (CAXIS.EQ.'R') THEN
    DO J=1,NY
        CALL TRANSLATE (DATA(1,J),NX,1,TX,TY,TZ,FILTER)
        DA=SXINC*AK0
        CALL FFT2 (- 1,NX,1,DA,DATA(1,J))
        IF (NPOL.EQ.2) THEN
            CALL TRANSLATE (DATA2(1,J),NX,1,TX,TY,TZ,FILTER)
            CALL FFT2 (- 1,NX,1,DA,DATA2(1,J))
        END IF
    END DO
ELSE
    DO I=1,NX
        DO J=1,NY
            SDATA(J)=DATA(I,J)
        END DO
        CALL TRANSLATE (SDATA,1,NY,TX,TY,TZ,FILTER)
        DA=SYINC*AK0
        CALL FFT2(- 1,1,NY,DA,SDATA)
        DO J=1,NY
            DATA(I,J)=SDATA(J)
        END DO
    END DO
    IF (NPOL.EQ. 2) THEN
        DO I=1,NX
            DO J=1,NY
                SDATA(J)=DATA2(I,J)
            END DO
            CALL TRANSLATE (SDATA,1,NY,TX,TY,TZ,FILTER)
            CALL FFT2(- 1,1,NY,DA,SDATA)
            DO J=1,NY
                DATA2(I,J)=SDATA(J)
            END DO
        END DO
    END IF
END IF

RETURN
END
```

\$CDS ON

FUNCTION SINX(X)

C

Program last revised: 20 OCT 1999

IF (ABS(X).GE.1.E-06) THEN

SINX=SIN(X)/X

ELSE

SINX=1.- X*X/6

END IF

RETURN

END

\$CDS ON

!
! This subroutine clears the terminal display.
Subroutine called: None

SUBROUTINE SWIPE

CHARACTER*4 A,G,U

A=CHAR(27)//'H'//CHAR(27)//'J'

!Clear Alpha display

G=CHAR(27)//'*da'

!Clear Graphics display

U=CHAR(27)//'&j@'

!Clear User Keys display

WRITE(1,5) A,G,U

FORMAT (3A4)

RETURN

END

\$CDS ON

SUBROUTINE TESTP2(N,ISP2)

C TESTS N FOR POWER OF TWO. IF N IS A POWER OF TWO,
C ISP2=0; IF NOT, ISP2=1.
C

XTRY=ALOG(FLOAT(N))/0.69314718
XDEL=XTRY-INT(XTRY+.001)
ISP2=0
IF(ABS(XDEL) .GT. 1.E- 5) ISP2=1
RETURN
END

\$CDS ON

SUBROUTINE TRANSLATE (DATA,NX,NY,X,T,Z,FILTER)

C Performs a translation of the data set in physical space using
C the vector R=(X,Y,Z). The data set domain is assumed to
C be K-space and the multiplier is $\exp(-j K.R)$.
C

C An ideal low-pass filter can also be applied. The FILTER
C parameter is a radius (in normalized wave-number units).
C Data points beyond this distance from the wave-number origin
C are zeroed. A value of FILTER=0. implies no filtering.

COMPLEX DATA(NX,NY), CFACT,CJ
CHARACTER CAXIS*1,POL*8,CSCAN*80,NAME*15
EMA DATA

COMMON /PARAM/ RSCAN(7),CAXIS,POL,CSCAN,NAME,IDATE(3),ITIME(3)
COMMON /WVGE/ A,B,AK0

CJ=0.,1.)
XKINC=RSCAN(2)*AK0 !X axis spacing
YKINC=RSCAN(5)*AK0 !Y axis spacing
XK0=RSCAN(1)*AK0 !Initial X-axis point
YK0=RSCAN(4)*AK0 !Initial Y-axis point

IF (FILTER.EQ.0) FILTER=1. !SAME AS NO FILTER
R2=(FILTER*AK0)**2 !Filter radius

DO J=1,NY
 YK=YK0+(J-1)*YKINC
 YK2=YK**2
 DO I=1,NX
 XK=XK0+(I-1)*XKINC
 XK2=XK**2
 IF ((XK2+YK2) .GT. R2) THEN
 DATA(I,J)=(0.,0.)
 ELSE
 ZK2=AK0**2- XK2- YK2
 CFACT=CEXP(- CJ*(X*XK+Y*YK))
 IF (ZK2 .GT. 0.) THEN
 ZK=- SQRT(ZK2)
 CFACT=CFACT*CEXP(- CJ*Z*ZK)
 ELSE IF (ZK2 .LT. 0.) THEN
 ZK=- SQRT(- ZK2)
 CFACT=CFACT*EXP(Z*ZK)
 END IF
 DATA(I,J)=DATA(I,J)*CFACT
 END IF
 END DO
END DO

RETURN
END

\$CDS ON

SUBROUTINE XYTHUY(UX,UY,SX,SY,SK,CK,SA,SB)

C CONVERTS X,Y COMPONENTS OF TRANSFORMED SPECTRUM TO HUYGENS
C COMPONENTS IN ORTHOGONAL DIRECTIONS A AND B.

COMPLEX SX,SY,SZ,SA,SB

ST=SQRT(UX**2+UY**2)

CT=SQRT(1.-ST**2)

SZ=- (UX*SX+UY*SY)/CT

IF (ST .LT. .0001) THEN

HBX=CK

HBY=SK

HBZ=0.

HAX=- SK

HAY=CK

HAZ=0.

ELSE

CP=UX/ST

SP=UY/ST

CPB=CK*CP+SK*SP

SPB=- SK*CP+CK*SP

CPA=- SPA

SPA=CPA

C This is Huygens unit polarization pattern for X electric field.

HX=SPB**2+CPB**2*CT

HY=SPB*CPB*(CT- 1.)

HZ=- CPB*ST

HBX=CK*HX- SK*HY

HBY=SK*HX+CK*HY

HBZ=HZ

HX=SPA**2+CPA**2*CT

HY=SPA*CPA*(CT- 1.)

HZ=- CPA*ST

HAX=- SK*HX- CK*HY

HAY=CK*HX- SK*HY

HAZ=HZ

END IF

SA=SX*HAX+SY*HAY+SZ*HAZ

SB=SX*HBX+SY*HBY+SZ*HBZ

RETURN

END

\$CDS ON

SUBROUTINE XYTYCON (UX,UY,SX,SY,SPOL,CPOL,SEL,SAZ)

C Converts X,Y components of transformed spectrum (Sx,Sy) to azimuth,
C elevation components (conical about Y-axis) including a possible
C rotation about the Z-axis by angle POLOUT, where

CPOL=COS(POLOUT)

SPOL=SIN(POLOUT)

C
C Components are computed for a direction Ux,Uy.

COMPLEX SX,SY,SAZ,SEL,SZ,GAM,CB,SA,CA,CSQRT

GAM=CSQRT(CMPLX(1.- UX*UX- UY*UY,0.0))

SZ=- (UX*SX+UY*SY)/GAM

SB=UY

! SIN EL

CB=CSQRT(CMPLX(1.- SB*SB,0.0)) ! COS EL

SA=UX/CB

! SIN AZ

CA=GAM/CB

! COS AZ

SEL=((CPOL*(- SB*SA)+SPOL*CB)*SX+(SPOL*SB*SA+CPOL*CB)*SY+(- SB*CA)*SZ)*GAM

SAZ=((CA*CPOL*SX)- (CA*SPOL*SY)- (SA*SZ))*GAM

RETURN

END

\$CDS ON

SUBROUTINE XYTZCON (UX,UY,SX,SY,SPOL,CPOL,S10TH,S10PH)

C
C Converts X,Y components of transformed spectrum (Sx,Sy) to spherical
C components (theta, phi - conical about Z- axis) including a possible
C rotation about the Z- axis by angle POLOUT, where

CPOL=cos(POLOUT)

SPOL=sin(POLOUT)

C

COMPLEX SX,SY,S10TH,S10PH,SZ,GAM,CTH,STH,CPH,SPH

GAM=CSQRT(CMPLX(1.- UX*UX- UY*UY,0.0))

SZ=- (UX*SX+UY*SY)/GAM

CTH=GAM

! COS THETA

STH=CSQRT(1.- GAM*GAM)

! SIN THETA

SPH=UY/STH

! SIN PHI

CPH=UX/STH

! COS PHI

S10TH=CTH*(CPH*CPOL- SPH*SPOL)*SX+CTH*(SPH*SPOL+CPH*SPOL)*SY- STH*SZ

S10PH=-(CPH*CPOL- SPH*SPOL)*SY- (SPH*CPOL+CPH*SPOL)*SX

RETURN

END

\$CDS ON

```
SUBROUTINE XYZOPEN(FNAME,IUNIT,ISTATUS)

CHARACTER CAXIS*1,POL*8,CSCAN*80,NAME*15,INFILE*25,STAT*7,FNAME*15
COMMON /RECBUFF/LBUF(8200)
COMMON /PARAM/RSCAN(7),CAXIS,POL,CSCAN,NAME,IDATE(3),ITIME(3)
COMMON /USER/IWRITE,IREAD

C      XYZOPEN open a datafile.
C      LGBUF is a library subroutine to enlarge I/O buffer size. NOTE:
C      the buffer array LBUF must not be in EMA Under any circumstances.
C      NOTE: if CDS id used, then either the call to LGBUF must be made in
C      the main program(in this case common block RECBUFF is not required),
C      or common block RECBUFF must be declared in the main program and
C      this subroutine. If CDS is not used then the call can be made from
C      this subrotine without using common block RECBUFF.

NAME=FNAME
GOTO 77

5      WRITE(IWRITE,*) 'Enter data file name:'
      READ (IREAD,20) NAME
20     FORMAT(A)
77     INFILE=NAME//'.':XYZFILE'

      IF (ISTATUS .EQ. 0) STAT='OLD'
      IF (ISTATUS .EQ. 1) STAT='NEW'
      IF (ISTATUS .EQ. 2) STAT='UNKOWN'

      IF (STAT .EQ. 'NEW') THEN
          NPTS=RSCAN(6)
          IF (CAXIS .EQ. 'X') NPTS=RSCAN(3)
          IRECLB=(NPTS*4)+2      ! RECORD LENGTH(BYTES)-- AMP OR PHASE AND STATUS
          IF (IRECLB .LT. 180) IRECLB=180      !INSURE ENOUGH ROOM FOR HEADER REC.
          CALL DATETIME(IDATE,ITIME)
      ELSE
          INQUIRE(FILE=INFILE,IOSTAT=IERR,ERR=65,RECL=IRECLB)      !READ RECORD LTH
      END IF

      OPEN(UNIT=IUNIT,FILE=INFILE,ACCESS='DIRECT',FORM='UNFORMATTED',RECL=IRECLB,IOSTAT=IERR,STATUS=STAT)

10     FORMAT('ERRO ON OPENING FILE',A15)

65     IF (IERR .GT. 0) THEN
          WRITE(IWRITE,10) NAME
          GOTO 5
      ELSE
          CALL LGBUF(LBUF,IRECLB/2)      !ENLARGE I/O BUFFER TO #BYTES/2
      END IF
      RETURN
END
```