APPENDIX F. PROGRAM LISTINGS.

The listings below were obtained during compilation on a PDP11/10 minicomputer. FORTRAN line numbers added during the compilation are shown at the left of each FORTRAN statement. These numbers can be used in reporting any problems to the author. Columns 73 to 80 of many lines have been used for a short comment indicating the purpose of the line, and these are not part of the program code.

The compilations shown were obtained under FORTRAN 4. The programs have been run without modification on some systems using FORTRAN 77. With other machines a few changes may be required. The 'constants' given in DATA statements must retain their values between successive calls to the subroutines. This is not mandatory in FORTRAN, and with some versions of FORTRAN 77 the DATA must be changed to SAVE or STATIC. For systems which carry out run-time checking of array bounds, most of the DIMENSION statements must be changed from FV(40), HT(40) to FV(8), HT(8). Some compilers may also require that statements of the type: DO ## I=1, N+5 be replaced by: N5= N+5 / DO ## I=1,N5.

In the listings below, program statements which are a necessary part of the calculation are shown in upper case. Lines in lower case relate to ouput listings or trace facilities, and can be ignored for purposes of following the basic program logic.

F.1 THE MAIN SUBROUTINE POLAN

```
      ------------------------------------------------------------------------
0001          SUBROUTINE POLAN (N,FV,HT, FB,DIP,START, AMODE,VALLEY,LIST)
      c
      c   - -  Generalised POLynomial real height ANalysis  - -        may73/feb84.
      c  Overlapping, Single or Least Squares,  with full start and valley options.
      c        If problems arise, run data with  list = 4  and mail all output to:-
      c        J.E.Titheridge,  Physics Dept.,  University of Auckland, New Zealand.
      cxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
      c                      INPUT  DATA                                          *
      c                                                                           *
      c--- Polan is called with frequency, height data in the arrays  fv, ht.     *
      c                                                                           *
      c--- Initially:-  set  n  equal to the dimension of the arrays  fv, ht;     *
      c this must be greater than  30 + the number of data points in the arrays.  *
      c                                                                           *
      c Intermediate layers terminate at a scaled critical frequency (or fc = 0.) *
      c        with:  h' = 0  for a chapman peak and normal valley,               *
      c               h' = 10 for a peak  with no following valley,               *
      c               h' = negative for a cusp-type discontinuity only.           *
      c    The o-ray fc (scaled or zero) may be followed by an x-ray value (-fcx). *
      c                                                                           *
      c The final layer is terminated by at least 2 null points, with  h = f = 0. *
      c Data can be terminated without a peak by using a final frequency of -1.0. *
      cxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
      c                      NORMAL  INPUT  PARAMETERS                            *
      c                                                                           *
      c--- FB   gives the gyrofrequency at the ground in MHz, for inverse cube.   *
      c                                                                           *
      c--- DIP  is the magnetic dip angle in degrees.                             *
      c                                                                           *
      c--- START normally gives a model height at 0.5 mhz.  Typical values are:   *
      c   noon   sunset-2/rise+2hr   set/rise    set+1hr   set+2   set+4 to rise-1 *
      c   85km    88km(E layer)    90(E)/80(F)   100 km    130 km     150 km.     *
      c                                                                           *
      c-   An extrapolated starting height is used if  start = 0.                 *
      c-   With initial  x-ray  values a slab start correction is calculated from *
      c              0.3*fmin  (adding points at 0.3, 0.6 and 0.8 *fmin).         *
      c-   (With x-ray data,  start gives the gyrofrequency height for underlying *
      c              ionisation;  the values listed above are still suitable.)    *
      c---------------------------------------------------------------------------*
      c---      The final three parameters are zero,   for most work.            *
      c                                                                           *
      c--AMODE sets the type of analysis, as listed below.    Zero uses mode 5.   *
      c        Use amode+10 for 12-point integrals, for high accuracy at large    *
      c        dip angles (this is done automatically, at dip.ge.60, when amode=0).*
      c     Values of amode greater than 29 are used to specify the number of     *
```

```
c        polynomial constants to be used to describe each ionospheric layer; *
c     e.g. 80 uses an 8-term real-height polynomial for each separate layer, *
c          85 uses 8 terms for the final layer and 5 terms for lower layers. *
c                                                                             *
c--VALLEY= 0.0 or 1.0  to use the initial default width of twice the local   *
c     scale height.  The initial default depth is  0.05 MHz.              *
c        The calculated depth is scaled according to (calculated width)**2.  *
c--LIST  = 0  prints results for the start, peak and valley regions only.    *
c        = 1  shows the frequ range and polynomial coefficients at each step.*
c        = 2  adds some start/peak output; 3,4 add more detail for each step.*
cxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
c                       RETURNED  DATA                                        *
c                                                                             *
c  POLAN returns with  frequencies, real heights  in the arrays  fv, ht.     *
c   N = no of data points returned,  with peak at  fc=fv(n-3), hmax=ht(n-3). *
c         Points at  n-2,n-1 and n are extrapolated heights at 0.5, 1.0 and  *
c                                        1.5 scale heights above the peak.    *
c   fv(n+1) gives the standard error of the critical frequency,  in mhz.      *
c   ht(n+1) gives the standard error of the peak height,  in km.              *
c   fv(n+2) gives the total electron content to the peak in  e10/m**2.        *
c   ht(n+2) gives the scale height of the peak in km;                         *
c   (a negative value shows an unreasonable peak extrapn was limited.)       *
cxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
c                       CHANGES TO NORMAL OPERATION                           *
c  FB   negative to  use a constant gyrofrequency  fh = -fb.                  *
c  DIP  negative to omit physical checks on the calculated profile segments.  *
c  START between 0. and 44. defines the plasma frequency for a model start.   *
c  start = -1.0  to use a direct start, from the first scaled point.          *
c  start < -1.0  for x-starts to use a polynomial from (-start -1.0) mhz.     *
c  AMODE negative to omit physical relations (c3.3) from start/valley calcns.*
c  VALLEY = 10.0  for a monotonic  (no valley)  analysis.                     *
c  valley =  5.0  for a maximum-valley (upper reasonable limit) analysis.     *
c  valley =  0.1  to  5.0  multiplies the standard valley by this factor.     *
c  valley = -.01  to -.99  to use  -valley  as the initial depth,             *
c                          (instead of the default value 0.05 mhz).           *
c  valley = -1.0  to  iterate both valley depth and width for best fit;       *
c              (-1.x to iterate from an initial depth of 0.x mhz ).           *
c  valley = -2.01 to -30 specifies a fixed valley width of 5*int(-valley) km;*
c                 and any decimal part of valley specifies the depth in mhz.  *
c  LIST = 5 shows each set of simultaneous equations, in the matrix  b(i,j).  *
c         6/7/8/9 give more detail for the start/reduction/peak/valley step.  *
c                 Use list negative to trace only the peak and valley steps.  *
c                       (-1 gives output for the first starting step also).   *
c           List= -10 suppresses all output, even the normal layer summaries.*
cxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
c                       STANDARD MODES OF ANALYSIS:                           *
c Amode=1.- Linear Lamination analysis    (with chapman peaks, and valleys).  *
c Amode=2.- Parabolic Lamination analysis, matching end gradients  (= paul).  *
c Amode=3.- Overlapping Cubics, with no spurious oscillations(JATP 1982 p657)*
c Amode=4.- Fourth Order Overlapping Polynomials (Radio Science 1967 p1169).  *
c Amode=5.- Fifth Order Least Squares fit to 6 points  (4 virtual + 2 real).  *
c Amode=6.- Sixth Order Least Squares fit to 8 points  (5 virtual + 3 real).  *
c Amode=7.- Sixth Order  fitting  7  virtual + 3 real heights;  calculate 2.  *
c Amode=8.- Sixth Order  fitting  8  virtual + 4 real heights;  calculate 2.  *
c Amode=9.- Seventh Order fitting 13 virtual + 6 real heights;  calculate 3.  *
c Amode=10. A Single Polynomial,  fitting  0.73(nv+2)  terms to  nv heights.  *
c Amode= 10L,  where L is an integer in the range 3 to 15, uses a single     *
c              polynomial with L terms to describe each ionospheric layer.    *
c Amode= 10L+M  uses  L terms for the final layer, and M for earlier layers.  *
cxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
c
c  In the program listing, error checking, debug
c  and output statements are in lower case.
c                   Normal printed outputs from Polan are indicated by  ------->
c                   Trace outputs and abnormal conditions are shown by  **----->
c                   Fuller debug outputs obtained with  list > 0        ##----->
c                   Loops are delimited by  c.....
c-------------------------------------------------------------------------------
0002        DIMENSION  FV(40), HT(40), IT(20),IV(20),IR(20),IH(20)
0003        COMMON /POL/ B(40,17),Q(18), FH,ADIP, MODE,MOD, FA,HA, TCONT,LBUG
```

```
0004        COMMON /POL/ HS, FC,FCC, SH, PARHT, HVAL,VWIDTH,VDEPTH, XWAT
0005        COMMON /POL/ MAXB,NF, NR,NL, NX, MS,MT,JM, LK, KR,KRM, KV,MF, NC  counters
      c
      c              !------ first step -------!  !---- following steps -----!
      c  At MODE =  1 2  3  4  5  6  7  8  9 10  1  2  3  4  5  6  7  8  9 10
0006        DATA IT/1,2, 3, 4, 4, 5, 6, 6, 7,73, 1, 2, 3, 4, 5, 6, 6, 6, 7,73/  nterms
0007        DATA IV/1,2, 3, 4, 5, 7, 8,10,12,35, 1, 1, 2, 3, 4, 5, 7, 8,13,35/  nvirts
0008        DATA IR/0,0, 0, 1, 1, 2, 2, 3, 5, 2, 0,-1,-1, 1,-2,-3,-3,-4,-6,-3/ realhts
0009        DATA IH/1,1, 2, 3, 3, 4, 5, 6, 8,28, 1, 1, 1, 1, 1, 1, 2, 2, 3,28/ calchts
      c
0010            tcontf(x) = x**j*(fa*(fa/j+2.*x/(j+1))+x*x/(j+2))            polycont
0011            maxb= 40                                                     B rows
      c         maxb= row dimension of array B.  Set IR(10 and 20) to maxb-5
      c-------------------------------------------------------------------------------
      c$$$$$$$$$$$$$$$$$$$$$$$$$$$ (1) Initialisation  $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
      c1 ---                        (1) Initialisation
      c
0012        if(list.gt.1)print10,n,fv(1),ht(1),fb,dip,start,amode,valley,list
0014  10         format ('0#ARGS: N,f1,h1='I4,2F7.2,5X,'fb,dip,start=',3F7.2,
           1             5X,'amode,valley,list=',2F7.2, I4 /)
      c1 ---
      c1.1                          Set trace, gyrofreq  and  mode  constants.
0015        ADIP= ABS(DIP)
0016        FH  = GIND(FB,-ADIP)                                             groundfh
0017           lbug= list
0018           if (lbug.gt.1) call trace (fv,ht, 0.)                         ##----->
0020        MODE = ABS(AMODE)
0021           IF (MODE.EQ.0.and.ADIP.GE.60)  MODE = 15
0023           IF (MODE.EQ.0)  MODE = 5
0025        MOD2 = MODE - MODE/10*10
0026        MOD1 = MOD2
0027        IF (MODE.EQ.10.or.MODE.GE.30)  MOD1 = 10                         snglpoly
0029        FV(N-1) = -1.                                                    end data
      c-------------------------------------------------------------------------------
      c1.2--  Identify the start method to use, and set the following parameters:-
      c       fa,ha = the origin (starting point) of the first real-height polynomial;
      c       lk = 1 / 0 / -1   for   o-ray / poly / slab start.
      c  Move virtual height data to start at  fv(31), ht(31).   Virtual, real height
      c  origins are at  kr, kv  =  1, 31-js  (where js=no of freqs added below fmin).
      c       Add an interpolated point at fv(30),ht(30) to control o-ray starts.
      c
0030        CALL  SETUP (N, FV, HT,  START)
      c
0031        DHS = AMIN1(HS-HA,0.)                                            for c3.3
      c$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
      c
      c////////////   R E A L   H E I G H T   A N A L Y S I S   L O O P
      c
      c For each cycle:   Calculate one polynomial, with nt terms, from the point **
      c         fa=fv(k), ha=ht(k) to fit the next nv virtual and nr real hts.    **
      c                     (nr negative to include one real height below ha).    **
      c       Calculate a further nh real heights,  and set  k = k+nh.           **
      c                                                                          **
      c   If a critical frequency is found in k+1 to k+nv+1, calculate up to     **
      c         the preceding freq, and determine a least-squares chapman peak.  **
      c With x-ray data (-ve freqs) at start or after critical, recalculate ha  * ** *
      c                                                                          ****
      c Real height origin (fa,ha) is at k=kr;  virtual at k=kv.  krm=top real.  **
      c*****************************************************************************
      c
      c$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ (2) Prepare  Data $$$$$$$$$$$$$$$$$$$$$$$$$
      c2 ---                            (2) Prepare  Data
      c
      c2.1                               Start of analysis (or restart after a peak)
0032  200   MOD = MOD1                                                       =1 to 10
0033        KRM = KR                                                         top real
0034        NNR = 0
0035        GO TO 230
      c2.1a                                  Later segments (using mod+10)
0036  210   IF (MOD.GT.10)  GO TO 220
```

```
0038          KR = KR-IR(MOD)                                                  step bak
0039          KV = KV-IR(MOD)
0040          MOD = MOD+10
0041  220   NNR = IR(MOD)                                                      real hts
      c2.1b                                  All segments
0042  230   NT = IT(MOD)                                                        terms
0043        NV = IV(MOD)                                                     virtuals
0044        NH = IH(MOD)                                                      to calc
0045        NR = IABS(NNR)
0046        NL = MINO(1,NR-NNR)                                               prev ht
0047        MS = 0
0048          IF (LK.EQ.1.AND.KV.EQ.29) MS = 1                               GradEqun
0050          IF (NV.GT.1.AND.NV.LT.8)  NV = NV - MS                         at start
      c-------------------------------------------------------------------------------
      c2.2                  Count initial x-rays. Check frequency sequencing.
      c                     Check for cusp, peak, or end of data.
      c Set nf= no of o-rays (=nv, if sufficient points exist before a peak/restart);
      c     nx= x-rays;   mv= nf+nx;   fm= fv(mf) = top frequency in this step.
      c     fcc = fc or 0.1 for a peak,  = -.1 for cusp at fm,  = 0. otherwise.
      c
0052        CALL  SELDAT  (NV, FV, HT)
      c
0053                      tras= -2.2
0054                      if (nv.lt.0)  go to 630                          exit>>>>
0056        FM = FV(MF)                                                    top freq
0057        MV = MF-KV                                                     #  freqs
0058        IF (NF.GT.NV)  NH = NH + (NF-NV)/2                             to calc.
0060              if (list.lt.-3.and.list.ge.-9) lbug= 1                    summary
0062              if((fcc.gt.0..or.fc.gt.0.).and.list.ge.-9) lbug=iabs(list) trace on
0064           if(lbug.gt.2.or.(lbug.eq.2.and.mod.lt.10))calltrace(fv,ht,2.2) ##----->
0066        IF (KR.EQ.1)  GO TO 300                                           start
      c-------------------------------------------------------------------------------
      c2.3    Subtract the group retardation due to the last calculated real-height
      c   section.  This modifies all h' at f > fa (where fa = fv(kr)),  and increases
      c   the index lk (giving the height to which group retardation is removed) to kr
      c
0068        CALL  REDUCE (FV, HT)
      c
0069                      tras = -2.3
0070                      if (jm.le.0) go to 640                           exit>>>>
0072                                if (lbug.gt.2) call trace (fv,ht,2.3)   ##----->
      c-------------------------------------------------------------------------------
      c$$$$$$$$$$$$$$$$$$$$$$$$$ (3) Set Up Equations for next profile step $$$$$$$$
      c3 ---                    (3) Set Up Equations for next profile step
      c
      c3.1                                  Initialise mt,fa,fc and counters.
0074  300   MT = NT
0075        IF (NT.GT.20)  MT = (NF+2) *NT/100                               mode 10
0077        IF (NX.GT.0)   MT = MT + (NX+1) /2                              + x-rays
0079        IF (NF.GT.NV)  MT = MT + (NF-NV)/3
0081        IF (MODE.LT.30)  GO TO 310
0083          MT = MODE/10                                                   setterms
0084          IF (FV(MF+2).NE.0..and.MOD2.GE.2.and.NF.LT.NV)  MT = MOD2      not last
0086  310   MT = MINO(15,MT,MV+NR+MS)                                          terms
0087        JM = MT
0088        FA = FV(KR)                                                    newstart
0089        HA = HT(KR)                                                    newstart
0090        NC = 0                                                         iteratns
      c                                     Check for valley, set valley flag hval
      c                                     Initialise valley width and depth
0091        LOOP = 0
0092        CALL STAVAL (FV,HT, START,VALLEY, LOOP)
      c
      c                                     Is this a start or valley calcn?
0093        IF (LK.GT.0.and.HVAL.EQ.0.)  GO TO 320                         not valy
0095          IF (MODE.GE.30)  MT= MT-1                                    fixedJM
0097          MT = MINO(MT,MV-1)                                           polterms
0098          JM = MT+1                                                    + offset
0099  320                             if (lbug.gt.3) call trace (fv,ht,3.1) ##----->
      c-------------------------------------------------------------------------------
```

```
      c3.2                                  Set Up Equations in B
      c
0101  330   CALL COEFIC (MV,FV,HT)                                          setcoefs
0102                  tras = -3.2
0103                  if (jm.le.0)  go to 640                              exit>>>>
0105        IF (AMODE.LT.0..OR.MV+4.GT.MAXB)  GO TO 380                     no phys
0107        IF (MS.NE.1) GO TO 340
0109              G = 1.0 + 1.8/FV(30)                                     o start,
0110              B(MV+1, 1)  = .5                                           set Q1
0111              B(MV+1,JM+1)= .5*G*(HT(30)-HA)
0112  340   IF (LK.GE.0)  GO TO 360
      c3.3                                  (add ms physical start relations. x only)
0114              WS = 0.1                                                   weight
0115              IF (START.GT.44.)  WS = 0.5
0117           B(MV+1,JM)   = WS
0118           B(MV+1,JM+1) = DHS*1.6 *WS                                 q(jm)=dh
0119           B(MV+2,MT)   = WS*.3
0120           B(MV+2,JM+1) = (HS/3.-20.) *WS*.3                             slab
0121           B(MV+3,MT-1) = WS*.3                                       loworder
0122           MS = 3
0123           GO TO 380
      c
0124  360   IF (HVAL.EQ.0.)  GO TO 380
      c                                   (add ms physical valley relations. x or o)
0126              WV = 1.0                                                   weight
0127              IF (HVAL.LT.-2.)  WV = 10.                               fix valy
0129              IF (NX.GT.0)  WV = 0.2
0131           B(MV+1,JM)   = WV
0132           B(MV+1,JM+1) = (VWIDTH - PARHT) *WV                        stddvaly
0133           B(MV+2,MT) = 0.5                                           loworder
0134           B(MV+3,1)  = 0.4                                            grad-
0135           B(MV+3,JM) = -.1/VDEPTH                                      -cont.
0136           B(MV+4,MT-1) = 0.15                                        loworder
0137           MS = MINO(4, MT)
      c
0138  380   NC = NC+1
0139            if(lbug.gt.2.or.(ms.gt.0.and.lbug.ne.0)) call trace(fv,ht,3.3)##----->
      c$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$  (4) Real-Height Solution $$$$$$$$$$$$$$$$$$$$
      c4 ---                              (4) Real-Height Solution
      c4.1                                    Solve equations in B
0141        NS = MINO( MV+NR+MS, maxb)
0142        CALL  SOLVE (NS, JM, B, Q, DEVN)                                 calc. q
      c
      c4.2                                    Check physical constraints
      c
0143                            if (ms .eq. 1) lbug= lbug+200                leave q1
0145                            if (dip.lt.0.) lbug= lbug+400                noadjust
0147        Q(18) = DEVN
0148        CALL ADJUST (HA,FA,FM,FC, LK,JM,MT, B,Q, LBUG)
      c
      c
      c4.3                                    Iterate a valley or x-start calcn.
0149        LOOP = 2
0150        IF (JM.EQ.MT.or.NX.GT.0)  GO TO 420
      c                                        0-ray Valley; loop once to adjust depth
0152           VWIDTH = Q(JM) + PARHT
0153           IF (NC.EQ.1.and.HVAL.NE.0.)  LOOP = 1                        adj.vdep
0155  420        if(lbug.gt.1.and.hval.ne.0..or.lbug.gt.2)calltrace(fv,ht,4.3)##----->
      c
0157        IF (FB.LT.0.)  LOOP = -IABS(LOOP)                               fixed fh
0159        IF (JM.GT.MT)  CALL STAVAL (FV,HT, START,VALLEY, LOOP)          strt/val
0161        IF (LOOP.EQ.4)  GO TO 330                                       valyloop
0163        IF (LOOP.EQ.3.and.FB.GT.0.)  GO TO 330                          xrayloop
      c$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ (5) Store Real Heights $$$$$$$$$$$$$$$$$$$$
      c5 ---                              (5) Store Real Heights
      c
0165        KRM= KR+NR-NL                                                   lastreal
0166        KVM= KV+NR-NL
0167        NH = MINO(NH,NF-MOD1/10*2)                                      to calc.
0168        MQ = MT+MINO(LK,0)                                              polterms
```

```
0169        QM = Q(MQ)
0170        KA = KVM + 1                                                    next pt.
      c.....
0171        DO 520  K = KA, MF
0172           FR = FV(K)
0173           KRM = KRM+1
0174           DELTF = FR-FA
0175           FV(KRM) = FR
0176  520      HT(KRM) = HA + SUMVAL(MQ,Q,DELTF,1)
      c.....
0177        FN = FV(KVM+NH)
0178        DF = FV(KVM) - FA
0179        IF (FCC.EQ.0.)  GO TO 540
0181           NH = KRM-KR                                                  to peak.
0182           FN = FM
0183  540   DO 560  J = 1, MQ
0184  560      TCONT = TCONT+J*Q(J)*(TCONTF(FN-FA) -TCONTF(DF))             polycont
      c
0185        KR = KR+NH                                                      step on
0186        KV = KV+NH                                                      origin
0187        FC = FCC
0188        IF (FC.EQ. 0.)  GO TO 210                                       do next
0190        IF (FC.EQ.-.1)  GO TO 200                                         cusp
      c$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ End  of  Normal  Steps  $$$$$$$$$$$$$$$$
      c$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ (6) Calculate and List Peak data $$$$$$$$$$$$
      c6 ---                              (6) Calculate and List Peak data
      c
      c6 ---                    Least-squares fitting of a Chapman-layer peak.
      c
0192                tras= 6.
0193                if (fc.lt.0.)  go to 620                                 end>>>
0195                if (ht(krm).lt.90.)  go to 640                         abort>>>
0197        CALL  PEAK  (FV, HT,  HMAX)
      c-------------------------------------------------------------------------------
      c7 ---                              (7)  Restart for a New Layer,  or end.
      c7.1
0198        HS = (HMAX + HT(KRM))/2.                                          for fh
0199        IF (HT(KV+1).GT.0.)  GO TO 200                                  newlayer
      c
      c7.2                  All finished.  Add points at  z = .5, 1.0, 1.5  above peak,
      c                                        using a scale height gradient of 0.1.
0201        N = KR + 3
0202        FV(N-2) = FC *.97372
0203        FV(N-1) = FC *.91213
0204        FV(N  ) = FC *.83462
0205        HT(N-2) = HMAX + SH *0.5128
0206        HT(N-1) = HMAX + SH *1.0526
0207        HT(N  ) = HMAX + SH *1.6216
0208        FV(N+2) = TCONT*.124                                              t.e.c.
0209        HT(N+2) = SH                                                    scale ht
0210        FV(N+3) = 0.
0211        HT(N+3) = 0.
0212        RETURN
      c-------------------------------------------------------------------------------
      c7.3                                    E x i t   o n   E r r o r
0213  620           if (fc.eq.-1.)  go to 660
0215  630           if (nv.lt.0)  kv = -nv -1                               >>seldat
0217  640           if(lbug.ge.-9)print650, (fv(i),ht(i),i=kv-1,kv+3),tras,kv **----->
0219  650                       format ('0>>>>>error at f,h ='10f8.3,
     $                                ' section',f4.1,i5,'  >>>> end', /)
      c
0220                                    call trace (fv,ht,tras)               **----->
0221  660           n = kr + 1
0222                fv(n) = fv(kv+1)
0223                ht(n) = ht(kv+1)
0224                fv(n+1) = 0.
0225                ht(n+1) = 0.
0226                return
0227        END
```

F.2 THE SUBROUTINES SETUP, SELDAT AND STAVAL

Subroutines called by POLAN for preparation of input data for successive steps of the real-height calculation are grouped in a file POLSIN.FOR. The first subroutine SETUP is called only once near the beginning of POLAN to determine basic start constants. These include the smallest values of plasma frequency and virtual height in the data, the start method to be used, and an appropriate extrapolated value for the starting height (used in extrapolated starts, and to define the gyrofrequency in the low-density underlying region). The given data are also moved up to start at positions 31 in the frequency, height arrays, so that calculated real-heights can be filled in from the beginning of the same arrays.

SELDAT is called at the beginning of each new step in POLAN. It determines which data points will be used to calculate the next real-height polynomial, and deletes any unwanted extraordinary ray data.

STAVAL supervises all start and valley calculations within POLAN. It is always called during the initialising phase for each real-height step, to determine whether a valley is involved and to set the valley flag HVAL accordingly. If the following calculation is to determine a valley, initial estimates for the valley width and depth are made, based on a model value for the neutral scale height, and flags are set to control the type and path of the valley iteration.

After a real-height solution involving a start or valley, STAVAL is called again to check the result. Calculations involving extraordinary ray data and a height-varying gyrofrequency are iterated, at least once, adjusting the value of gyrofrequency to match the last-calculated real heights. Valley calculations are also iterated to keep the assumed value of valley depth in a fixed relation to the calculated width. For two-parameter valley calculations an additional iteration loop is required to determine independent values of depth and width to give the smallest r.m.s. error in the fit to the virtual-height data (and to the physically-desirable conditions included in the valley calculation).

```
      c  POLSIN.FOR  =  SETUP,  SELDAT,  STAVAL.
      c**************************************  -  input processing for Polan.
      c
0001              SUBROUTINE   SETUP  (N, FV,HT,  START)
      c ---                                (called from section c1.2 of polan)  feb84.
      c 1. Move the virtual height data up 30 places, to start at  fv(31), ht(31).
      c 2.  Set  N = number of data points (up to two zero heights).
      c       fmin =  the lowest plasma frequency in the data.
      c       hmin =  the lowest virtual height in the data.
      c          (these are used for start extrapolation in the absence of x-ray data).
      c 3. Identify the start method, and set the following parameters:
      c       js = the number of points added below fmin, = 2/1/0 for oray/xray/direct.
      c       fa, ha  define the starting point for the first polynomial.
      c       hs =  the height to evaluate the gyrofrequency for an x-ray start.
      c       lk = 1 / 0 / -1   for   o-ray / x-ray poly / x-ray slab   starts.
      c The real and virtual origins are at  kr = 1  and  kv = 31-js.
      c-------------------------------------------------------------------------------
      Constants which SETUP sets in /pol/ are:-  lk,kr,kv, fa,ha, hs,vdepth.
      c
0002        DIMENSION  FV(40), HT(40)
0003        COMMON /POL/ B(40,17),Q(18), FH,ADIP, MODE,MOD, FA,HA, TCONT,LBUG
0004        COMMON /POL/ HS, FC,FCC, SH, PARHT, HVAL,VWIDTH,VDEPTH, XWAT
0005        COMMON /POL/ MAXB,NF, NR,NL, NX, MS,MT,JM, LK, KR,KRM, KV,MF, NC  counters
0006        VDEPTH = 0.
0007        VWIDTH = 0.
0008        TCONT = 0.
0009        SH = 0.
0010        FC = 0.
      c1.a --                          Move virtual data to start at  fv(31), ht(31)
      c1.a --                          Find index j of the 1st o ray; fmin= lowest fn
0011        K = N-30                                                          max data
0012  10       FV(K+30)= FV(K)
0013           HT(K+30)= HT(K)
0014              IF (FV(K).GT.0.)  J= K
0016           K = K-1
0017           IF (K.GE.1)  GO TO 10
0019        FMIN = FV(J)                                                      1st oray
0020        IF (J.GT.1)  FMIN = AMIN1(FMIN,SQRT(FV(1)*(FV(1)+.9*FH)))        1st fn
```

```
      c1.b --
      c1.b --                    Set starting conditions (direct, extrap or model).
      c                             Calculate initial fa, ha;- start from fmin:
0022        HMIN = HT(J)
0023           DO 20 I = 1, 5
0024  20       IF (HT(J+I).GT.45.)  HMIN = AMIN1(HMIN, HT(J+I))                  h'min
0026        HA = HMIN
0027        FA = FMIN
0028        HS = HA
0029        JS = 0
0030        LK = 1
0031        IF (START.EQ.-1.0.AND.J.EQ.1)  GO TO 40                            direct
      c                                                          - start extrapn:
0033        JS = 2                                                            add 2pts
0034        DH = ABS(HT(J+2)-HT(J)) *FMIN /(FV(J+2)-FMIN)                     to f = 0
0035        HS = AMIN1(HMIN-DH, HMIN/2.+50.)                                  start fh
0036        HS = AMAX1(HS, HMIN/4.+55.)                                       lowlimit
0037        FA = AMIN1(.5, .6*FMIN)
      c                                                          - model  height:
0038        IF (ABS(START).GE.45.) HS = AMIN1( ABS(START), HS*.4+HMIN*.6 )      · model
0040          IF (J.GT.1)  GO TO 30                                             x start
0042        HA = HS                                                           start ht
0043        IF (MODE.EQ.10 .OR. MODE.GE.20)  FA = FMIN*.6                     snglpoly
0045        IF (START.GE.45..OR.START.EQ.0.) GO TO 40                          o start
      c                                                          - model density:
0047        H = INT(START*.1)*10
0048        HA = 90.+H+H
0049        FA = START-H
0050        GO TO 40
      c                                                          - x-ray calculn:
0051  30      JS = 1                                                          add 1pt.
0052          HA = HS*.4 + HMIN*.6                                             x start
0053            LK = 0
0054            IF (START.GT.-3.)  FA = -START-1.                             polystrt
0056            IF (START.LT.-1.)  GO TO 40
0058          LK = -1
0059          FA = FMIN*.6                                                    slabstrt
0060          VDEPTH = FMIN*.3
      c1.c --
      c1.c --          Store initial points. Virtual data starts at index kv (29 to 31).
0061  40    KV = 31-JS
0062        FV(30) = (FA + FMIN)/2.0                                          o start,
0063        HT(30) = HMIN - DH*(FMIN-FV(30))/FMIN                             add virt
0064                 if (j.eq.1)  hs= ht(30)                                  (& show)
0066        FV(KV) = FA
0067        HT(KV) = HA
0068        FV(1) = FA
0069        HT(1) = HA
0070        KR = 1                                                            realstrt
0071        RETURN
0072        END


      cc*****************************************************************************
0001        SUBROUTINE  SELDAT  (NV, FV, HT)
      c                                  called from section c2.2 of polan.    feb84.
      c     Select the virtual-height data to be used in the next step of polan.
      c     Define the following parameters:-
      c  nf = number of o-ray data points to be used
      c       (equal to nv, if sufficient good points exist before a peak/restart);
      c  nx = number of x-rays points;
      c  fm = fv(mf) = highest frequency included in this step;
      c  fcc is set equal to zero for a normal (non-peak) step;
      c             equal to  -.1 for  a cusp at fm;
      c             equal to a scaled critical frequency (fc),  or to 0.1, for a peak.
      c-----------------------------------------------------------------------------
0002        DIMENSION  FV(40), HT(40)
0003        COMMON /POL/ B(40,17),Q(18), FH,ADIP, MODE,MOD, FA,HA, TCONT,LBUG
0004        COMMON /POL/ HS, FC,FCC, SH, PARHT, HVAL,VWIDTH,VDEPTH, XWAT
0005        COMMON /POL/ MAXB,NF, NR,NL, NX, MS,MT,JM, LK, KR,KRM, KV,MF, NC  counters
```

```
      c
      c   ffit gives the minimum desirable frequency range, for fitting x-ray data.
      c   gfit gives the maximum slope  dh'/df  which should be used in an x-start.
0006        data   ffit,  gfit    /  0.4, 40. /
      c-------------------------------------------------------------------------------
      c2.a                      Count initial x rays. Check frequency sequencing.
      c2.a                      Check for cusp, peak, or end of data.
0007        FCC = 0.                                                          no peak
0008        FSX = 0.
0009        FRX = 0.
0010        MF = KV
0011        FS = FV(KV)                                                       start f
0012        F1 = 0.
0013        FH = GIND(0., HT(KR))
0014        NF = 0
      c.....                                  Frequency loop  (nf = 1 to nv):
0015  10    NF = NF+1
0016  12       MF = MF+1                                                      toppoint
0017           FM = FV(MF)                                                    top freq
0018           FN = FV(MF+1)                                                  nextfreq
0019           HV = HT(MF)                                                    top virt
0020                          if(lbug.eq.6.and.kr.eq.1)print *,nv,nf, f1,fm   ##----->
0022           IF (FM.GT.FS)  GO TO 20
0024              FRX = SQRT(FM*(FM+FH))                                      top xray
0025              IF (MF.EQ.KV+1)  FSX=FRX                                    1st xray
0027              IF (FM.LT.0..AND.FRX.GE.FSX.AND.NF.EQ.1)  GO TO 12           x ray.
0029  15                nv = -mf
0030                    return                                                error>>>
0031  20          IF (F1.EQ.0.)  F1 = FM
0033              IF (HV.LT.0..OR.FN.LT.0.)  GO TO 40                         cusp,end
0035              IF (ABS(HT(MF+1)).LE.30.)  GO TO 50                             peak
0037           FS = FM
0038                    if (fn.le.fm)  go to 15                               error>>>
0040        IF (NF+1.LT.NV.AND.MF-KV.LT.MAXB-4)  GO TO 10
      c.....
      c                                       Check final point
0042             GRAD = (HT(MF+1)-HV) / (FN-FM)
0043        IF (NF+1.EQ.NV.and.(NF.LE.2.or.KR.GT.1.or.GRAD.LT.100.)) GO TO 10 loopfreq
      c
      c                            Leave loop, with nf = no of o rays (.le.nv).
      c2.b
      c2.b                                    Check need for additional points.
0045        IF (MOD.GT.10.OR.MOD.LT.4)  GO TO 70                            end freq
0047           IF (HT(MF+1)-HV.GT.(FN-FM)*GFIT)  GO TO 70                   retarded
0049           IF (FRX.GT.FSX  .AND. FM.GT.FRX)  GO TO 70                   all x in
0051           IF (FM-F1+.04*(nf-nv).LT.FFIT .OR.FM.LT.FSX)  GO TO 10        incr fm
0053        GO TO 70
      c                                       Cusp  (fcc =-0.1) or end data
0054  40          FCC = AMIN1(FN, -.1)                                      discont.
0055              HT(MF) = ABS(HV)                                           (=cusp)
0056              GO TO 70
      c                                       Peak  (fcc = 0.1  or scaled fc)
0057  50                FCC = AMAX1(FN,.1)                                   f crit.
      c
      c2.c       Shift data arrays to delete out-of-range x rays (fnr > fm+0.1).
      c2.c
0058  70    MX = MF - NF                                                      lastxray
0059  80    NX = MX - KV                                                       x rays
0060        IF (NX.EQ.0)  GO TO 100
0062           F = FV(MX)
0063           IF (SQRT((F+FH)*F).GT.FM+0.1) GO TO 90                         delete
0065           IF (NX.EQ.1)  GO TO 100
0067           IF (HT(MX)-HT(MX-1).LT.GFIT*(FV(MX-1)-F))  GO TO 100          grad ok.
0069  90          J = MX
0070  95             FV(J)= FV(J-1)                                          move up1
0071                 HT(J)= HT(J-1)
0072                 J = J-1
0073                 IF (J.GE.KV)  GO TO 95
0075              KV = KV+1
0076              GO TO 80
```

```
0077  100   IF (NX.EQ.0.AND.KR.EQ.1)  LK = 1                                  no xstrt
0079        RETURN
0080        END


      cc*************************************************************************
0001        SUBROUTINE STAVAL (FV,HT, START,VALLEY, LOOP)
      c                                                                       feb84.
      c       Staval handles all start and valley calculations, in which the
      c       real-height polynomial has a constant term  [at q(jm), where jm=mt+1]
      c       so that the height at the origin  [fa,ha]  is recalculated.
      c-------------------------------------------------------------------------
      c--> The initialising call enters and exits with  loop = 0.
      c        If this is not a valley calculation (i.e.  hval = 0.)  then staval just
      c            sets the initial weight  xwat  for any x-ray data, and returns.
      c        For a valley calculation (shown by hval.ne.0.)
      c            the iteration-flags  devl, devll  are initialised,  and
      c            the standard values for valley depth and width are calculated
      c                          (based on the 'standard scale height'  sha).
      c
      c--> A call with  loop = 1  is to adjust the valley depth for a single o-ray
      c                          iteration.  this exits with  loop = 4.
      c-------------------------------------------------------------------------
      c--> The main call enters with  loop = 2.            ( -2 if fb is negative. )
      c       Exit with  loop = 2  shows that the start/valley calculation is completed
      c                          and the required real heights have been added to ht.
      c       If  loop > 2  on exit,  polan recycles the valley calculation:-
      c           loop = 3  requests a new calculation with a changed gyrofrequency.
      c           loop = 4  is to recalculate with a new value of valley depth  vdepth.
      c-------------------------------------------------------------------------
0002        DIMENSION  FV(40), HT(40)
0003        COMMON /POL/ B(40,17),Q(18), FH,ADIP, MODE,MOD, FA,HA, TCONT,LBUG
0004        COMMON /POL/ HS, FC,FCC, SH, PARHT, HVAL,VWIDTH,VDEPTH, XWAT
0005        COMMON /POL/ MAXB,NF, NR,NL, NX, MS,MT,JM, LK, KR,KRM, KV,MF, NC  counters
      constants:    base,    depthfac,   depthconst
0006        DATA    VBASE,    VDEEP,       VCONST     / 0.6, .008, 20. /
      constants:  1st depth,  scale,  peak-top increase          for valley iteration
0007        DATA    VAL1,     DVAL1,       VPEAK      / .1001, 6., 1.4 /
      constants:   xweight,    fhht itern limit
0008        DATA     WVX,          HXERR              / 1.0,  2.0 /
      c to remember between calls:                    (next DATA to SAVE for fortran77)
0009        DATA  VAL, DVAL, DEVL, DEVLL, HMAX, HDEC, FHHT  /  7* 0.0  /      to store
0010        sq(x) = sqrt((1.-x)*(1.+x))
      c-------------------------------------------------------------------------
0011            NFB  = LOOP                                                   fix fb ?
0012            LOOP = IABS(LOOP)
0013        IF (LOOP.GT.1)  GO TO 200
0015        IF (LOOP.EQ.1)  LOOP = 4                                          o ray
0017        IF (LOOP.EQ.4)  GO TO 40
      c#########################################################################
      cx1---                      First call (with loop = 0) from polan section c3.
0019            tras = -3.1
0020        XWAT = WVX
0021        PARHT= 0.
      cx1.a                                Check for valley, set flag hval
0022        HVAL = HT(KV)                                                      valley
0023            IF (HVAL.EQ. 0.) HVAL = VALLEY
0025            IF (HVAL.EQ. 0.) HVAL = 1.
0027            IF (HVAL.GE.10.) HVAL = 0.                                  novalley
0029        IF (HVAL.EQ.0.)  RETURN                                         novalley
      c-------------------------------------------------------------------------
      c                                  Initialise valley width and depth
0031        HDEC = INT(HVAL) - HVAL
0032        DEVL = 1.E6
0033        DEVLL= 1.E7
0034        DVAL = DVAL1                                                    valscale
0035        HMAX = HT(KR)
0036        SHA = HMAX*.25-20.                                              scale ht
0037        HS = HMAX + SHA                                                 vally fh
```

```
0038        SH = SH *VPEAK                                                  topside
      cx1.b                       Set standard width
0039  20      VWIDTH = SHA*2.  * ABS(HVAL)                                  standard
0040              IF (HVAL.LT.-2.)  VWIDTH = INT(-HVAL) *5.                 specifid
      c-------------------------------------------------------------------
      c                   Set initial depth,  or depth adjustment for valley iteration
      cx1.c                       Standard depth (width calculations iterate here).
      c
0042  40      VAL = VWIDTH**2 *VDEEP/(VCONST+VWIDTH)                        v. depth
0043            IF (HDEC.GT.0.)  VAL = HDEC                                 specifid
0045              IF (HVAL.EQ.-1..AND.NX.GT.0)  VAL = VAL1                  1stdepth
0047              IF (INT(HVAL).EQ.-1.)  HVAL = -1.                          iterate
      c
      cx1.d                       Peak section  (depth calculations iterate here).
0049  60      VDEPTH = VAL*FA/(VAL+FA)                                      depth<fa
0050            PARHT = 2.0*SH* SQ(1.-VDEPTH/FA)                            par.vall
0051            HA = HMAX + PARHT                                           valy bot
0052            IF (NC.EQ.0)  FHHT = HA+20.                                 field ht
0054        RETURN
      c##########################################################################
      c##########################################################################
      c                   Main call  with  loop = 2,  from polan section c4.3.
      c                              Start / valley  entry,  after  "solve".
0055  200   tras = -4.3
0056        DEVN = Q(18)
0057        FNL = 0.
0058        DHL = Q(JM)
      c
      cx2--                                   Calculate and check real heights
0059  220   IF (NX.LE.0..OR.(DEVN.EQ.0..AND.HVAL.EQ.-1.)) DVAL = -1.          list,end
0061        QM = Q(JM)
0062        HR = HA + QM
0063        HN = HR
0064        FN = FA
0065        MC = NC
0066        MV = NF + NX
0067        MQ = MT + MINO(LK,0)
      c.....                                    (real hts stored for coefic recycle)
0068        DO 300  I = 1, MV
0069           HL = HN
0070           FL = FN
0071           FN = FV(KV+I)
0072           IF (FN.LT.0.)  FN = SQRT((FN+FH)*FN)
0074           DELTF = FN - FA
0075           HN = HR + SUMVAL(MQ,Q,DELTF,1)
      c
0076              if (nc+mc.gt.25.or.start.eq.-.1) go to 300                 omittest
0078              if ( (hn-hl)/(fn-fl) .gt. 2.0)   go to 300                 grad ok.
      c                                                 dh / dfn  .lt.  2.0
0080              if (nx.gt.0.and.xwat.eq.wvx.and.nc.gt.1) go to 380             redo
0082                 a = qm-amax1(5.,abs(qm)/4.)*amax0(1,3-kr)
0083                 nqm = int(a*10.01)                                      next qm
0084                 if (fn.ne.fnl.or.hn-hl.gt.dhl) go to 350                lower qm
0086                    if (lbug.ge.-9) print 250, fv(kv+i)                  **----->
0088  250               format('0data/gyrofrequency incompatible at f ='f6.2)
0089                    mc = 25                                             noadjust
0090  300      IF (KR+I.LT.KV)  HT(KR+I) = HN
      c.....
0092        KRM = MINO(KR+MV, KV-1)                                         topcalcd
      cx3---                                  Set field ht  (and recycle fb).
0093        IF (NC.LE.2)  XWAT = WVX
0095        KHX = KR + 1 + NX/3
0096        DHX = HT(KHX) - FHHT
0097        FHHT = HT(KHX)
0098           IF (NC.EQ.1 .OR. ABS(DHX).GT.HXERR)   LOOP = 3               reset fb
0100           IF (NX.LE.0.OR.NFB.LT.0.OR.NC.GE.20)  LOOP = 2               no reset
0102        IF (KR.GT.1)  GO TO 500                                           valley
0104        GO TO 400                                                        x start
      c-------------------------------------------------------------------------
      c              (jump here to get new solution with reduced start offset)
```

```
0105  350              dev = devn                                              decr. qm
0106                   call solve (nqm, -jm, b,q, devn)
0107                   if(lbug.gt.0) print360, nc, qm,q(jm), fn,hn, dev,devn  **----->
0109  360        format(i3,' staval: qm reduced from'f6.1,' to'f6.1,', to avoid
     $ -ve slope at f,h ='2f7.2,6x,'(devn increases'f6.2,' to'f6.2,').')
0110                   fnl = fn
0111                   dhl = hn-hl
0112                   go to 220
      c-------------------------------------------------------------------------
      c                                        Re-calculate a bad result
0113  380                    xwat = wvx/2.                                      bad data
0114                         if(lbug.ge.-9)print 390                           **----->
0116  390                    format (' x ray weights reduced to 1/4.')
0117                         loop = 4
0118                         return
      c*************************************************************************
      cx4---****************************** X ray Start: slab (.3-.6f1) +poly********
      cx4---
0119  400    SLAB = Q(MT)                                                      polystrt
0120             IF (LK.EQ.0)  SLAB = 0.
0122             if(lbug.ge.-9)print 420, nc,qm, slab,devn, jm,nf,nx,ms,fhht -------->
0124  420        format (i3,'  start offset =',f6.1,' km,   slab',f6.1,
     1                ' km.',6x,'devn',f6.2,' km',i5,' terms fitting',
     2               i3,' 0 +',i2,' X rays +',i2,1h.,6x,'hx =',f6.1)
0125         IF (LOOP.EQ.3)  RETURN                                           reset fb
0127         KD = 1-LK                                                        addedpts
0128         GO TO 700
      c*************************************************************************
      cx5---****************************** Valley:  Iterate and print        ********
      cx5---
0129  500    VWIDTH = QM + PARHT
0130               if (lbug.gt.0.or.dval.lt.0..or.nc.ge.20)
     1            print 520, nc, vwidth,vdepth, devn, jm,nf,nx,ms, fhht     ------->
0132  520          format ( i3,'  valley',f5.1,' km wide,',f8.2,' MHz deep.'
     1                 ,7x,'devn',f6.2,' km',  i9,' terms fitting',
     2                 i3,' 0 +',i2,' X rays  +'i2,10x,'hx =',f6.1)
0133         IF (LOOP.EQ.3) RETURN                                            reset fb
      c
0135         KD = 4
0136         LOOP = 4
0137         VBOT = QM*VBASE
0138         SLAB = QM-VBOT                                                   lin.slab
0139            IF (DVAL.LT.0..OR.NC.GT.25)  GO TO 700                        end valy
0141            IF (HVAL.EQ.-1.)  GO TO 600                                    iterate
      c
      c - - - - - - - - - - - - - Normal cycle: Scale depth as (width)**2  - - - - - - -
0143         IF (DEVN.GT.DEVL/.9.and.nc.lt.4)  GO TO 20                       revert
0145         IF (DEVL.LT.1.E6)  GO TO 560                                     end valy
0147             DEVL = DEVN
0148             GO TO 40                                                     loopvaly
0149  560    DVAL = -1.
0150             if (lbug.eq.0)  go to 500                                    listvaly
0152         GO TO 700                                                        end valy
      cx6--- - - - - - - - Vdepth Iteration: choose depth val1 or val1*dval1. - - - -
      cx6---
0153  600      IF (DVAL.NE.DVAL1) GO TO 620                                    iterate
0155           IF (VAL.EQ.VAL1)   GO TO 675                                    1stround
0157           DVAL = 1.+.5*COS(.015*DIP)                                      2ndround
0158           IF (DEVN.LT.DEVL)  GO TO 675                                    continue
0160             VAL = VAL1                                                    revert
0161             GO TO 680
      c                                      Iterate from chosen depth
0162  620        IF (DEVN.GT.DEVL*.97-.003.AND.DEVLL.LT.1.E6)  GO TO 660        end
0164             IF (DEVN.LT.DEVL)   GO TO 670                                 continue
0166               DVAL = 1./DVAL                                              reverse,
0167               VAL = VAL*DVAL                                              2ndentry
0168               DEVLL = DEVN
0169               GO TO 680
      c - - - - - - - - - - - -              Interpolate to minimum deviation
0170  660        DMIN = (DEVLL-DEVN)/ABS(DEVLL+DEVN-DEVL-DEVL)*.5  - 1.
```

```
0171            DVAL = -DVAL**AMIN1(DMIN, .5)
0172            GO TO 680                                                    last
      c                                    Recalculate valley width
0173  670     DEVLL= DEVL
0174  675     DEVL = DEVN
0175  680   VAL = VAL*ABS(DVAL)                                              next
0176        GO TO 60
      c-------------------------------------------------------------------------
      cx7---*****************************  Tidy up.    Delete x rays.   ***********
      cx7---              Add initial points;   kd = 1,2,4  for poly, slab, valy
0177  700   LOOP = 2                                                         all done
0178                        if (lbug.eq.6) call trace (fv,ht, tras)          ##----->
0180        KR = KR + KD                                                     stepreal
0181        KV = KV + IABS(NX)                                               delete x
0182        IF (KD.EQ.0)  RETURN
0184        IF (KD.EQ.1)  GO TO 750                                          polystrt
      c                                    Store valley,start slab at kd = 4,2
0186          FBOT = FA - VDEPTH
0187          HT(KR-2) = HA                                                  v.bottom
0188          FV(KR-2) = FBOT                                                v.bottom
0189          HT(KR-1) = HA+QM*VBASE                                         v.bottom
0190          FV(KR-1) = FV(KR-2)                                            v.bottom
0191          IF (KD.LT.4)  GO TO 750
0193              HT(KR-3) = HA - 0.5*PARHT                                  peak top
0194              FV(KR-3) = 0.5*SQRT(3.*FA**2 + FBOT**2)                    peak top
0195              FV(KV) = FA
      c
0196  750   HA = HA + QM
0197        IF (KD.LE.2)  KR = KD
0199        HT(KR) = HA
0200        FV(KR) = FA
0201        IF (KR.GT.3)  GO TO 800                                          notstart
      c                                    Add points on start polynomial
0203            HT(1) = HA-SLAB
0204            FV(KV-1) = FA                                                calc ht
0205            F1 = FV(KV+1)                                                1st freq
0206              IF (LK.EQ.0)  FV(KV-1) = (FA+F1)*.5                        (poly)
0208            FV(KV) = (FV(KV-1)+F1)*.5                                    calc ht
0209            KV = KV-2-LK
0210            VBOT = 0.
0211  800   IF(KR.GT.1) TCONT=TCONT+FV(KR-1)**2*(VBOT+.5*SLAB)+FA**2*.5*SLAB  lin.vall
           1                  + PARHT*(FA*FA-VDEPTH*(FA+FV(KR-1))/3.)        par.vall
0213                        if (lbug.eq.6) call trace (fv,ht,tras)           ##----->
0215        RETURN
0216        END
```

F.3 THE PROGRAMS COEFIC, ADJUST and REDUCE

COEFIC and REDUCE are the major subroutines required by POLAN for the work of real-height analysis. Along with the subroutine ADJUST which carries out checks on the result, they make up the file POLSUB.FOR.

The subroutine REDUCE is called first in each real-height cycle. Its purpose is to calculate the group retardation caused by the last section of the real height profile, at all higher frequencies. This retardation is subtracted from the virtual heights which remain to be analysed, so that the stepwise calculation of real-height segments can continue without further allowance for the presence of underlying ionisation. For a frequency range of about 0.5 to 1.0 MHz above the current origin (the starting point for the next real-height section), the virtual heights contain a large component due to group retardation in the last real-height section. Group retardations over this range are therefore calculated by integration of the full polynomial real-height expression for the last step. At higher frequencies the group retardation is smaller and is obtained with sufficient accuracy by a simplified procedure which requires only one value of group index per step but effectively allows for parabolic real-height segments. This simplification is made to speed the analysis. As in most real-height procedures, using reasonably large data sets, most of the time is occupied in calculating values of the group refractive index to use in this reduction step.

The subroutine COEFIC sets up the simultaneous equations which are used to define the polynomial real-height coefficients. These equations occupy successive rows in the array B(40,17). If MV is the number of virtual heights included in this step of the analysis, the first MV rows of B contain the virtual-height coefficients relating the real-height coefficients q(1), q(2), .. q(JM) to the values of h"(f) - HA at each frequency. h" is the 'reduced' virtual height, equal to the observed height less the group delay due to ionisation below the height HA, and H is the starting point for the current step in the real-height analysis. Calculation of these virtual-height coefficients uses 5, 12 or 17-point Gaussian integration, as described in section 5 and appendix B.3. Suitable weights are also placed on the coefficients in B, to give a graded weighting to the equations at different frequencies.

COEFIC also places coefficients which define known real heights (in terms of the polynomial coefficients) in the following NR rows of the array B, so that the new solution will include a least-squares fit to NR of the previously determined heights. For valley or start calculations POLAN places MS additional physically-derived conditions into following rows of B, before finding the solution for the complete set of MV+NR+MS simultaneous equations.

COEFIC is also used by the subroutine REDUCE to perform the accurate reduction of virtual heights, using the full polynomial expression for the previous section of the real height profile to calculate and remove the group retardation due to this section at higher frequencies. This reduction is carried out for MV frequencies above the current real-height origin (FA,HA) when COEFIC is called with the first parameter equal to -MV.

The subroutine ADJUST is called by POLAN immediately after the real-height solution is obtained, to carry out various checks on this solution and see if it is physically acceptable. If not, an additional constraint is added into the set of simultaneous equations and a new least-squares solution obtained. This process is carried out rapidly by the subroutine SOLVE, and gives a new (increased) value for the virtual-height fitting error. For adjustments which involve only the correction of non-physical variations in 'unseen' regions of the profile (at plasma frequencies less than the minimum observed frequency f_{min}), the adjustment is revoked if it causes an undesirably large decrease in the accuracy with which the calculated profile fits the virtual height data.

```
      c  POLSUB.FOR  =  COEFIC,  ADJUST,  REDUCE.
      c*****************************************
      c
0001        SUBROUTINE  COEFIC  (MV, FV, HT)
      c                                                          jan77.
      c   Calculate coefficients  b(i,j),  for the real-height polynomial
      c            h-ha = q(j)*(f-fa)**j;     mt terms,  fa=fv(k),  ha=ht(k).
      c   The first mv rows of b give virtual ht coefs at freqs k+1 to k+mv;
      c       following nr rows give real height coefs at freqs k+1 to k+nr.
      c   Adds terms for a linear section at x ray start or above peak.
      c   Subtracts a parabolic segment (with s.h.=1.40*sh) above peak.
      c
      c   Called with mv -ve (from "reduce", in section 2.3) to reduce the next
      c   |mv|  virtual heights by the delay in the previous section fa to fv(k).
```

```
      c-------------------------------------------------------------------------------
0002        COMMON /POL/ B(40,17),Q(18), FH,ADIP, MODE,MOD, FA,HA, TCONT,LBUG
0003        COMMON /POL/ HS, FC,FCC, SH, PARHT, HVAL,VWIDTH,VDEPTH, XWAT
0004        COMMON /POL/ MAXB,NF, NR,NL, NX, MS,MT,JM, LK, KR,KRM, KV,MF, NC  counters
      c
0005        DIMENSION TR(17),W(17), GAUSS(17),FNR(17), FV(9),HT(9)
0006        DATA TR  /  .046910077,.23076534, .50     ,.76923466,.95308992,
           1 .009219683,.047941372,.11504866,.20634102,.31608425,.43738330,
           2  .56261670,.68391575,.79365898, .88495134,.95205863,.99078032 /
0007        DATA  W  /  .11846344,.23931434, .28444444,.23931434,.11846344,
           a  .02358767,.05346966,.08003917, .10158371,.11674627,.12457352,
           b  .12457352,.11674627,.10158371, .08003917,.05346966,.02358767 /
0008        DATA  VB, FTC, GTC  /  0.6,  0., 180. /
0009           sq(x)  =  sqrt((1.-x)*(1.+x))
      c-------------------------------------------------------------------------------
      cc1 ---
      cc1 ---                          (1).  Set integration order and weights
0010        IRA = 1                                                          use 5 pt
0011        IRB = 5
0012        F1 = FV(KV+1)
0013        IF ((MODE.LT.8.OR.MV.LT.0).AND.F1.GT.0.)  GO TO 1
0015        IRA = 6                                                          use 12pt
0016        IRB = 17
0017  1     NFF = IABS(MV)                                                    freqs.
      c
0018        KM = KV + NFF
0019        FW = FV(KM+1)                                                     for wv
0020        IF (FW.LT.1.)  FW = FV(KM)*2. - FV(KM-1)                         at fw=0
0022        I = KV + MINO(NFF, MAXO(MOD-15,2)+NR)
0023        DW = FW - FV(I)                                                  for wv=1
0024        DA = FV(KV+NR-NL) - FA                                           for wv=1
0025        KP = KR
0026        IF (PARHT.GT.0.)  KP = KR-1                                      end poly
      c
      c                                    Set gyrofrequency height
0028        HR = HA + 10.*AMINO(1,KR-1)                                      refln ht
0029        IF (KRM.GT.KR)  HR = HT(KR+1)                                     for fh.
0031        IF (KRM.EQ.KR)  FTC = FA*1.2
0033        IF (MV.LT.0)  GO TO 7
0035        IF (LK.GE.0.OR.KRM.EQ.KR)  GO TO 4
      c                                    Slab start;   set fh level.
0037        DELTF = 0.
0038        GRAD2 = Q(1) - GTC
0039        IF (GRAD2.LT.0.)  GO TO 3                                         normal
0041  2           grad1 = grad2
0042              deltf = deltf + .1*fa
0043              grad2 = sumval(mt-1,q,deltf,2) -gtc
0044              if (grad2.gt.0..and.deltf.lt.fa*.45)  go to 2
0046        DELTF = DELTF + .1*FA* AMIN1(GRAD2,0.) /(GRAD1-GRAD2)
0047  3     FTC = (FTC+FA+DELTF)*.5                                          average
      c                                    Clear matrix B
0048  4         DO 5  I = 1, MAXB
0049            DO 5  J = 1, 17
0050  5         B(I,J) = 0.
0051  7     CONTINUE
      c=============================================================================
      c#########################  Loop frequencies: i= 1 to nff normally,
      c                                             = 0 to nff to fit 1 back height
0052        I = 1 - NL
0053        IF (I.EQ.0.AND.MOD.NE.12)  I = -1                                prev ht
0055        IF (MV.LT.0)  I = 1
      c=============================================  Loop frequencies:
0057  10    IREAL = NFF + NL + MAXO(I,0)
0058        KVI = KV + I
0059        KRI = KR + I
      cc2 ---
      cc2 ---                          (2).  Frequency and weight
0060        F  = FV(KVI)
0061        HV = HT(KVI)
0062        FR2 = F*F
```

```
0063          IF (KRI.LE.KRM)  HR = HT(KRI)  *.8  +HT(KR+1)*.2                refln ht
0065          FH = GIND(0.,HR)                                                  set fh
0066            IF (F1.LT.0.)  FH = GIND(0.,HS)                               x calcn.
0068            IF (F.LT.0.)  FR2 = F*(F+FH)                                    x ray
0070        FR = SQRT(FR2)
      c                                    Weighting for least squares analysis
0071        WREAL = 10.
0072        IF (KVI.LT.KV)  WREAL = 3.                                        back ht.
0074            IF (I.LE.0)   GO TO 55                                        realonly
0076                      if (fr.le.fv(kr).or.fr.ge.fw.or.parht.gt.2.*sh
           $                        .or.fr.le.fc.or.nff+nr.gt.40)  jm = -jm    fatal>>>
      cc        if (jm.le.0)print*,is,nff,i,kvi,kri,f,hv,fw,fr,fc,parht          @
0078                                       if (jm.le.0)  return                 end>>>
0080        IF (MV.LT.0)  GO TO 12
0082        WVIRT = 1.
0083        IF(NR.GT.NL.AND.FCC.EQ.0.)WVIRT=SQRT(AMIN1((FR-FA)/DA,(FW-FR)/DW)) taper w
0085        IF (F.LT.0.)  WVIRT = XWAT
      c-------------------------------------------------------------------------------
      c                                    Integrate from t = tb (at f=fa) to ta
      cc3 ---
      cc3 ---                          (3).  Set integration range and type
0087  12    TB = SQ(FA/FR)                                                    lowerlim
0088        TA = 0.                                                           reflectn
0089        IF (MV.LT.0)  TA = SQ(FV(KP)/FR)                                  upperlim
0091        IF (MV.LT.0)  GO TO 20
0093           FTC = AMIN1(FTC,FA*.1+FR*.9)
0094           TC = SQ(FTC/FR)
0095           IF (F1.LT.0.)  GO TO 15                                        x calcn.
0097           TC = 0.39-.05/COS(.016*ADIP)
0098           IF (TB.LT.1.2*TC.OR.INT(ADIP)-2*IRA.LT.58)  GO TO 20           dip60,70
      c
      c                                    Extended (17-point) integration.
0100  15          IRA = 1
0101              IRB = 17
0102              TA  = TC                                                     sectn 1
0103  20    TD = TB - TA
      cc4 ---
      cc4 ---                          (4).  Retardation in start/peak section
0104        DEPAR = 0.                                                        peak sum
0105        DELIN = 0.                                                        slab sum
0106        IF (LK.GE.0.AND.PARHT.EQ.0.)  GO TO 35
0108           IF (SH.NE.0.)  DZ = .5*PARHT/SH                                z at fm
0110           GINDV = 0.
0111           DO 30 IR = 6, 17
0112              IF (LK.LT.0)  GO TO 28                                      lin only
0114                 TP = SQ(FC/FR*SQ(TR(IR)*DZ))
0115                 DEPAR = DEPAR + GIND(F,TP) *W(IR)                        pardelay
0116  28           GINDV = GIND(F,SQ((FA-TR(IR)*VDEPTH)/FR))
0117  30           DELIN = DELIN + GINDV*W(IR)                              lindelay
0118  35    CONTINUE
      cc5 ---
      cc5 ---                          (5).  Group index for polynomial terms
0119        DO 50 IR = IRA, IRB
0120           IF (IR.NE.6.OR.IRA.NE.1)  GO TO 45
0122              TD = TA                                                     sectn 2
0123              TA = 0.
0124              FH = GIND(0.,HR)                                           refln fh
0125              IF (F.GT.0.)  GO TO 45                                     not xray
0127                 FR2 = F*(F+FH)
0128                 FR = SQRT(FR2)                                          refln fr
0129                 TD = SQ(FTC/FR)                                         corr. tc
0130  45        T = TR(IR)*TD+TA
0131           FN = SQ(T)*FR
0132           GAUSS(IR) = GIND(F,T)*T*W(IR)/FN *TD*FR2
0133  50       FNR(IR) = FN - FA
      c-------------------------------------------------------------------------------
      cc6 ---                          (6).  Store coefficients in array B
      c
0134  55    DO 90 J = 1, JM
0135           RH = (FR-FA)**J                                               real ht.
```

```
0136          DPEAK = 0.
0137          SUM = 0.
0138             IF (KVI.EQ.KV.AND. J.EQ.1)  RH = 1.                          mode 2
0140             IF (I.LE.0)  GO TO 90                                      realonly
0142             IF (J.EQ.JM)  DPEAK = DEPAR*PARHT                          par.peak
0144          IF (J.LT.MT.OR.(J.EQ.MT.AND.LK.GE.0))  GO TO 60
      c                                                 Start or valley terms
0146                RH = J-MT
0147                IF (J.EQ.MT)  SUM = DELIN                                lin.slab
0149                IF (LK.GT.0)  SUM = aINDV*VB+DELIN*(1.-VB)               lin.valy
0151                GO TO 80
      c                                                 Polynomial terms
0152  60        DO 70 IR = IRA,IRB
0153             A = GAUSS(IR)
0154             GAUSS(IR) = A*FNR(IR)
0155  70         SUM = SUM+A
0156             SUM = J*SUM                                                   delay
      c                                                 Coefficients B(i,j).
0157  80        B(I,J) = (SUM+RH)*WVIRT                                      virtual
0158          IF (MV.LT.0) HT(KVI) = HT(KVI)-Q(J)*SIGN(SUM,HV)-SIGN(DPEAK,HV)  reduce
0160  90        IF (IREAL.LE.NFF+NR)  B(IREAL,J) = RH*WREAL                    real
      c
0162        IF (MV.LT.0)  GO TO 100
      c-----------------------------
      cc7 ---                          (7).  Store Virt/Real heights as r.h.s.
0164        IF (I.GT.0)  B(I,JM+1) = (HV-HA-DPEAK)*WVIRT                     V r.h.s.
      c
0166        IF (IREAL.GT.NFF+NR)  GO TO 100
0168           REAL = HT(KRI) - HA
0169              IF(MOD.EQ.12) REAL = Q(1)+2.*Q(2)*(FA-FV(KR-1))             gradient
0171           B(IREAL,JM+1) = REAL*WREAL                                     R r.h.s.
0172  100   I = MAXO(I,0)+1
0173        IF (I.LE.NFF)  GO TO 10                                          loopfreq
      c############################ End of frequency cycle ==========================
0175        RETURN
0176        END
      c*******************************************************************************
```

```
0001        SUBROUTINE  ADJUST (HA,FA, FM,FC, LK,JM,MT, BB, Q, LBUG)
      c                                                                  jan82/jun83.
      c Called from section c4.2 of Polan, after the initial call to Solve,
      c Adjust is used to condition the real-height polynomial  (in  df = f-fa)
      c         h = ha + df*q(1) + df**2 *q(2) + ... + df**mq *q(mq)
      c     by ensuring that
      c 1-  The gradient  dh/df  is positive at  f = fa.
      c     For an 0-ray start calculation, an initial gradient  Q1  of
      c                       greater than 80 km/MHz is reduced to  160.Q1/(80+Q1).
      c     For an X-ray slab start (lk =-1) the gradient is limited above 160 km.
      c 2-  The solution is well-defined:
      c               -  the last 3 coefficients do not alternate in sign, and
      c               -  these coefficients have absolute values less than 999.
      c     These conditions are omitted for the peak and x-ray start, when a large
      c                        gradient may be required at one end of the fit range.
      c 3-  For x start calculations:  the underlying slab is positive,
      c                                the initial height offset is negative,
      c                                the initial height (+offset) is > 60 km.
      c 4-  For a valley calculation:  the initial gradient  dh/df > sha,
      c                                the initial curvature  q(2) < -1.5,
      c                                the initial height offset is positive.
      c
      c Adjustments alter the least-squares coefficients  q(1)  to  q(jm),
      c     and the rms deviation of the fit  devn  (stored in q(18)).
      c-------------------------------------------------------------------------
      c     When  jm = mt,  the polynomial ends at  mq = mt.
      c-----
      c     jm = mt+1  and  lk.le.0  indicates an x start calculation.
      c         The poly then ends at  mq = mt-1,  and
      c         q(mt) = underlying slab thickness,  preferably positive;
```

```
      c         q(jm) = starting height offset,  which must be negative.
      c-----
      c     jm = mt+1  and  lk.gt.0  indicates a valley calculation.
      c         The poly then ends at  mq = mt,
      c         and  q(jm) = the valley width,   which must be positive.
      c-----------------------------------------------------------------------
      c     Adjustments are made by   Call  Solve (mm, -n, bb, q, devn).
      c  This modifies the least-squares solution by adding the constraint
      c  q(n) = mm/10.,  with a weight of 10.   Any constraints previously
      c  applied at this or larger values of n are first removed by  Solve.
      c-----------------------------------------------------------------------
0002        DIMENSION BB(2), Q(18)
      c
0003              lflag= (lbug+50)/100                                    omit adj
0004              lbug = lbug- lflag*100
      c                                 Set mq = no of poly terms,   and head
0005        DEVN = Q(18)
0006        MQ = MT
0007        IF (LK.LT.0)  MQ = MT-1
0009              if (lbug.eq.-6.and.lk.lt.2)  lbug= 96                     start
0011              if (lbug.gt.2.or.(lbug.gt.0.and.lk.lt.2)) print7,(m,m=2,mt) ------->
0013              if (lbug.gt.0.or.(lbug.eq.-1.and.lk.lt.2))
     1                 print9,'--- ',lk,jm,mt,ha,fa,fm,(q(i),i=1,jm),devn ------->
0015 7            format ('0*adjust-----  lk jm mt    ha     fa    fm',6x,
     1                 'q1',9i9/41x,10i9)
0016 9            format (' *adjust', a6,3i3, f8.2,2f6.2, 10f9.2 /43x,10f9.2)
0017            if (lflag.gt.0.or.jm.lt.4) return                         noadjust
      c-----------------------------------------------------------------------
      c(a)                            Initial gradient;  must exceed 1.5
0019        MINQ1 = 2
0020        IF (LK.GT.1)  GO TO 50
      c                                 0,X Start: limit gradient to 160,320.
0022              QMAX = 80.
0023              IF (LK.EQ.-1)  QMAX = 160.
0025              IF (Q(1).GT.QMAX)  MINQ1 = 2.*QMAX*Q(1) / (QMAX+Q(1))
0027              GO TO 60
      c                                   Valley:  gradient > sha
0028  50          IF (JM.GT.MT)  MINQ1 = INT(HA/4.-20.)
      c
0030  60    IF (Q(1).GE.FLOAT(MINQ1).AND.MINQ1.LT.85)  GO TO 80
0032              CALL SOLVE  (MINQ1*10, -1, BB, Q, DEVN)                 q(1) adj
0033            if(lbug.gt.0)print9,'q(1)',lk,jm,mt,ha,fa,fm,(q(i),i=1,jm),devn------->
      c
      c                                   Valley:  initial curvature < -1.5
0035  80    IF (JM.LE.MT.OR.LK.LE.0.OR.Q(2).LT.-1.5) GO TO 100
0037                CALL SOLVE  (-15, -2,  BB, Q, DEVN)                   q(2) adj
0038            if(lbug.gt.0)print9,'q(2)',lk,jm,mt,ha,fa,fm,(q(i),i=1,jm),devn------->
      c-----------------------------------------------------------------------
      c(b)                    Check for unnecessary high-order coefficients.
      c                 Reduce order if last 3 signs alternate, and sizes increasing,
      c                                       or if any size exceeds 999.
0040  100   QQ = Q(MQ)
0041        IF (MQ.LT.5.OR.ABS(QQ).LT.150. .OR.FC.GT.0..OR.LK.LE.0) GO TO 200     o.k.
0043          IF (QQ/Q(MQ-2).GT.2. .AND. QQ/Q(MQ-1).LT.-1.)  GO TO 150        fix it
0045          IF (ABS(QQ).LT.999..AND.ABS(Q(MQ-1)).LT.999.)  GO TO 200          o.k.
      c                               Add constraint  q(mq) = 0.,  and reduce mq.
0047  150           DEVN = -1.
0048                CALL SOLVE (0, -MQ, BB, Q, DEVN)
0049            if(lbug.gt.0)print9,'mq ',lk,jm,mt,ha,fa,fm,(q(i),i=1,jm),devn ------->
0051                MQ = MQ-1
0052                GO TO 100
0053  200   CONTINUE
      c-----------------------------------------------------------------------
      c(c)       S T A R T  and  V A L L E Y:   check sign, and limit offset term
      c
0054        DEVL = DEVN
0055        IF (JM.EQ.MT)  GO TO 500
0057        IF (LK.GT.0 )  GO TO 400
      c--------                                    Start:-  slab
0059        QM = Q(MT)
```

```
0060        NQT= 1
0061        IF (QM.GE.0.)  GO TO 350
0063  320            CALL SOLVE (NQT, -MT, BB, Q, DEVN)
0064           if(lbug.gt.0)print9,'slab',lk,jm,mt,ha,fa,fm,(q(i),i=1,jm),devn------->
0066        NQT= INT(QM*10.)
0067        IF (DEVN.GT.DEVL*1.25) GO TO 320                                  slab= qm
      c                                      Start:-  offset
0069  350       IOFST = -1
0070            IF (Q(JM).GT.0.)  GO TO 450
0072            IF (LK.GE.0.OR.HA+Q(JM).GT.60.)  GO TO 500
0074                      IOFST = 10.*(60.-HA)
0075                      GO TO 450
      c--------                              Valley
0076  400   IF (Q(JM).GE.0.)  GO TO 500
      c                                Correct initial real-height offset
0078                      IOFST = 1
0079  450                 CALL  SOLVE (IOFST, -JM, BB, Q, DEVN)
0080          if(lbug.gt.0)print9,'ofset',lk,jm,mt,ha,fa,fm,(q(i),i=1,jm),devn------->
      c-------------------------------------------------------------------------
0082  500   Q(18) = DEVN
0083                  if (lbug.eq.96)  lbug= -6
0085        RETURN
0086        END
      c*****************************************************************************


0001                    SUBROUTINE   REDUCE  (FV, HT)
      c                                                                      feb84.
      c   To reduce the remaining virtual heights by the group retardation in
      c       the last calculated real-height section  (from  fa  to  fv(kr)).
      c
      c   Modifies all h' at  f > fv(kr),  by subtracting the group retardation.
      c   Increases lk (the height to which group retardation is removed) to kr.
      c       Use  LIST = 7  to show details of the reduction steps.
      c-------------------------------------------------------------------------
0002        DIMENSION  FV(40), HT(40)
0003        COMMON /POL/ B(40,17),Q(18), FH,ADIP, MODE,MOD, FA,HA, TCONT,LBUG
0004        COMMON /POL/ HS, FC,FCC, SH, PARHT, HVAL,VWIDTH,VDEPTH, XWAT
0005        COMMON /POL/ MAXB,NF, NR,NL, NX, MS,MT,JM, LK, KR,KRM, KV,MF, NC  counters
0006     7  format (' reduce:',15f8.3)
      c-------------------------------------------------------------------------
      c2.a--       Reduction using full polynomial expression (at f<fm+mode/25).
0007        K = MF
0008        FRED = FV(MF) + .04*FLOAT(MODE)
0009  20      K = K+1                                                      nextvirt
0010          IF (ABS(HT(K)).GT.30..AND.FV(K).LE.FRED)  GO TO 20           reduce k
      c---
0012          KV1 = KV+1                                                   minreduc
0013          KVM = K -1                                                   maxreduc
0014                    if(lbug.eq.7)print7, (fv(j),j=kv1,kvm)             ##----->
0016                    if(lbug.eq.7)print7, (ht(j),j=kv1,kvm)             ##----->
0018        CALL COEFIC (KV-KVM, FV, HT)                                   prevpoly
0019                    if(lbug.eq.7)print7, (ht(j),j=kv1,kvm)             ##----->
0021                    if (jm.le.0)  return                               exit>>>>
      c----------                                 Error check on reduced virt hts.
0023                    do 40  k = kv1, kvm
0024  40                if (abs(ht(k)).lt.ha)  go to 50                     low ht.
0026            GO TO 60
      c                                  list bad point (h'<ha) and delete it
0027  50                  if(lbug.ge.-9) print 52, (fv(j),ht(j), j=k-1,k+1)  **----->
0029  52                  format ('0*****reduce: data error at f, h =',
          $                                  6f8.3,  i5,' end >>>>')
0030  55                fv(k)= fv(k-1)                                       move up
0031                    ht(k)= ht(k-1)
0032                    k= k-1
0033                    if (k.gt.kv-4)  go to 55
0035                    kv = kv1
      c-------------------------------------------------------------------------
      c2.b--       Approx reduction at f>fred for delay in fv(lk) (=fa) to fv(kr)
```

```
0036  60    MQ = MT + MINO(LK,0)                                              polterms
0037        LK = MAXO(LK,1)+1
0038        FAR = FV(KR)                                                      top freq
0039        G2 = Q(1)
      c.....                              For section  fv(l-1) to fv(l)
0040        DO 100 L = LK,KR                                                  loopslab
      c                                   :-  Get mean density, thickness
0041          FL = FV(L-1)
0042          FN = FV(L)
0043          AVN = (FL**2+FN*(FL+FN))/3.                                     mean f*f
0044          HA = HT(L)
0045          DH = HA - HT(L-1)
      c                                   :-  Correct for curvature
0046          DELTF = FN - FA
0047          G1 = G2
0048          IF (FN.NE.FC.AND.FN.GT.FA)  G2 = SUMVAL(MQ,Q,DELTF,2)             dh/df.
0050          IF (FN.EQ.FC .OR.FN.LT.FA)  AVN = AVN+FN*(FN-FL)/3.               peak
0052          DH = DH+(G2-G1)*(FL+FN)*(FN-FL)**2/(12.*AVN)                    correctn
0053          FH = GIND(0.,HA)                                                scale fh
      c                                   :-  Loop frequency to end
0054          K = KVM
      c---
0055  80      K = K+1                                                         loopfreq
0056            F = FV(K)
0057            IF (F.EQ.-1.)  GO TO 100                                      end freq
0059            FR = F
0060            HV = HT(K)
0061        IF (HV.EQ.0. .AND.HT(K+1).EQ.0.)  GO TO 100                       end freq
0063                if (abs(hv).le.30.)  go to 80                             skippeak
0065                if (f.lt.0.)  fr = sqrt((f+fh)*f)
0067                if (fr.lt.far)  go to 90                                  f error.
0069            TAV = SQRT(1.-AVN/FR**2)
0070            HT(K) = HV - GIND(F,TAV)*DH*SIGN(1.,HV)                         reduce
0071                    if(lbug.eq.7)print7, fl,fn,tav,dh, f,hv,ht(k)         ##----->
0073          GO TO 80                                                        loopfreq
      c---                                End freq loop.   Delete bad data and end
0074  90                if(lbug.ge.-9)print 52, (fv(j),ht(j), j=k-1,k+1), k   *******>
0076                    fv(k) = -1.                                           end data
      c---
0077  100   CONTINUE
      c.....
0078        LK = KR
0079        RETURN
0080        END
```

F.4. THE SUBROUTINES PEAK, TRACE, SOLVE, SUMVAL AND GIND

This group of five miscellaneous subroutines is contained in the file POLMIS.FOR.

The subroutine PEAK is called by POLAN only at the end of the real-height calculations for an ionospheric layer. The gradients at the last several frequencies are used in a least-squares analysis to determine the scale height and critical frequency of the best-fitting Chapman layer. The height of the peak is then found by fitting this layer to the last two calculated real heights. The method used in these calculations is described elsewhere (Titheridge, 1985).

The subroutine TRACE is used only to print additional information so that the operation of POLAN can be checked. In normal use, when the input parameter LIST is equal to 0 or 1, this subroutine is never called.

The subroutine SOLVE (M, N, B, Q, DEVN) plays an important part in POLAN. Normal operation is obtained when the input parameter N, giving the number of unknowns to be solved for, is positive. Equations specified in the first M rows of the matrix **B** are then solved and results returned as q(1) to q(N). When M is greater than N a least-squares solution is obtained, and the r.m.s. deviation of this solution is returned as DEVN. At M = N an exact solution is obtained, and DEVN = 0. In both cases equations of the form

$$b_1q_1 + b_2q_2 + ... + b_Nq_N = b_{N+1}$$

are stored with the coefficients b_1 to b_{N+1} in columns 1 to N+1 of the array **B**. Successive equations occupy successive rows in the array; thus the coefficients for the ith equation are B(i,1) to B(i,N+1). The first M rows of **B** contain coefficients for the M equations to be solved. These are converted to upper triangular form by a series of orthogonal Householder transformations. The rows N, N-1, ..., 1 are then solved in succession to obtain the parameters q_N, q_{N-1}, ..., q_1. This procedure has several advantages over the older method of forming the normal equations and solving them by Gaussian elimination. It is much more stable and gives accurate results for polynomials using 15 or more terms, whereas the normal equations become very ill-conditioned for polynomials of order greater than about 6. The calculations are quite fast, require no row or column interchanges, and can give directly the r.m.s. deviation of the fit. The solution is carried out in place requiring no additional storage. The original equations are destroyed, but the triangulated form retained in the array **B** is directly useful for further examination or adjustment of the results.

Revised solutions are obtained by SOLVE when it is called with a negative value for the parameter N. This indicates that an additional equation is to be added into the least-squares solution. The new equation is always $q_j = M/10$, where the index j is equal to -N and the constant M is given by the first parameter in the call to SOLVE. Addition of this equation to the previously triangulated set gives NP = NQ+1 equations (where NQ is the total number of coefficients q, as given in the first call to SOLVE). These NP equations are reduced to a new set of NQ upper triangular results, by application of Givens transformation. The new coefficients q, and the corresponding value of DEVN, are then obtained.

Before the new solution is obtained in SOLVE the subroutine checks the flags stored in the array ISETQ. These show any constraints previously placed on the coefficients q. Any previous constraint on the value q_j is first removed. This is necessary to avoid competition between different constraints, giving a result with an intermediate value of q_j and a large value for DEVN. Removal of a constraint is achieved by adding in the same equation again, with all coefficients multiplied by $(-1)^{0.5}$; this gives an effective weight of -1 which cancels the effect of the previous constraint. Any existing constraints on q_{j+1} to q_N are also removed. Constraints can then be tried successively, at decreasing values of j, with each trial showing results of only the last constraint. If it is desired to include two separate constraints (on different values of q_j) in the final solution, these constraints are applied in order of increasing j. In all cases the new solution is obtained directly from the previous solution and the stored values of the triangularised matrix $B_{i,j}$. Recalculation of group index integrals and of the original coefficients in B is not required, so modified solutions are obtained almost instantaneously.

The subprogram SUMVAL is used for evaluating various sums and products of arrays, as defined in the listing. This is used by POLAN to obtain the values of real height and gradient from the polynomial real-height coefficients q, and for some other operations.

The subroutine GIND provides values of the group refractive index as a function of the wave frequency F, and the parameter T which defines the ratio of plasma frequency to wave frequency. An initialising call with T negative is used to set the value of gyrofrequency at the ground (GH), and the magnetic dip angle. A height-varying gyrofrequency is normally assumed. Calling GIND with F = 0 scales internal constants to correspond to a height of -T km. These constants are organised to allow rapid calculation of the values of group index, since this calculation takes most of the time in

a real height analysis. If a fixed value of gyrofrequency is to be used at all heights, the initial value of GH is made negative. The equations used provide results for the ordinary ray when the frequency F is positive, and for the extraordinary ray when F is negative. They are designed to maintain full accuracy for all values of dip angle (from 0 to 90°0), and all values of T (from 0 to 1) for both rays. The derivation of the equations is given in appendix D.3.

```
      c  POLMIS.FOR  =  PEAK,  TRACE,  SOLVE,  SUMVAL,  GIND.
      c**************************************************** -  utilities for POLAN.
      c
0001            SUBROUTINE   PEAK  (FV,HT, HM)
      c*********************************************************************jan78/dec84.
      c  Calculate and list parameters for a Chapman-layer peak.
      c
      c  Called with KR, KV pointing at last calculated real height before peak.
      c  Get FC, SH  by a least-squares fit to the last NK calculated heights.
      c  Include any scaled  FC, FCX,   and iterate from a model scale height.
      c     LIST = 8 adds detailed trace outputs.
      c-------------------------------------------------------------------------------
0002        DIMENSION   FV(9), HT(9)
0003        COMMON /POL/ B(40,17),Q(18), FH,ADIP, MODE,MOD, FA,HA, TCONT,LBUG
0004        COMMON /POL/ HS, FC,FCC, SH, PARHT, HVAL,VWIDTH,VDEPTH, XWAT
0005        COMMON /POL/ MAXB,NF, NR,NL, NX, MS,MT,JM, LK, KR,KRM, KV,MF, NC  counters
0006            sq(x) = sqrt((1.-x)*(1.+x))
      c-------------------------------------------------------------------------------
      c                                     Set frequencies
0007        KR = KR + 1                                                      to peak
0008        KV = KV + 1                                                      to peak
0009        FM = FV(KRM)                                                      top ht
0010        HN = HT(KRM) *.999                                              ht at fm
0011        NK = MINO(NF*10/(NF+10), KR/3) + 2                              f to fit
0012        KA = KV -1 -NK
0013        SH = .27*HN - 21.                                               model SH
0014        HM = HN + .3*SH                                                   for fh
0015        FH = GIND(0., HM)
0016        NKA= NK
0017          IF (FC.GT..1)  NK = NK + 1                                    incl. fc
0019          FCX = ABS(FV(KV+1))
0020          HCX = HT(KV+1)
0021          IF (ABS(HCX).GT.30..OR.FCX.LT.FM+.5*FH)  GO TO 100            no x ray
0023             NK = NK + 1                                                incl fcx
0024             FV(KV+1)= FV(KV)                                           swap fc
0025             FV(KV) = FCX                                                and fcx
0026             HT(KV+1)= HT(KV)
0027             HT(KV) = HCX
0028             KV = KV + 1
0029  100     grad = .5/fv(ka)
0030          fv(kr) = fc
      c-------------------------------------------------------------------------------
      c.....                                Set up and solve peak equations
0031        DO 400 J = 1, 2                                                   iterate
      c...
0032        DO 300 I = 1, NK
0033           F = FV(KA+I)
0034              IF (F.EQ.FCX)  F = SQRT(FCX*(FCX - FH))                     x crit.
0036           W = 2.**(20.*(F-FM)/F)                                          weight
0037           T = 0.                                                            peak
0038           IF (F.GT.FM)  GO TO  260                                      f crit.
0040              DELTF = F-FA
0041              GL  = GRAD
0042              GRAD = 4./F/ AMAX1( SUMVAL(MT,Q,DELTF,2), 3.)              2dN/dh/N
0043                 IF (I.EQ.(NKA+1)/2)  GM = GRAD                             mean
0045                 IF (I.EQ.NKA)  GRAD = GRAD *.995                        hn corn.
0047              T = SH*GRAD
0048  260      B(I,1) = W                                                     *log(FC)
0049           B(I,6) = (ALOG(1.+T)-T)*.25                                       save
0050           B(I,2) = B(I,6) *W                                                * SH
0051           B(I,3) = ALOG(F) *W                                             r.h.s.
0052                    if(lbug.eq.8.and.j.eq.2)print *,f,grad,t,b(i,6),w      ##----->
0054  300   CONTINUE
```

```
      C...
0055                  ht(kr) = ht(kv)
0056                  fv(kr) = fc
0057               if(lbug.ge.3)print320,nk,nka,sh,(fv(k),ht(k),k=kr-nka-1,kr) ##----->
0059  320       format(' peak fit: nk,nka,sh='2i3,f6.1,2x,14f7.2,(/35x,14f7.2))
      C
0060        CALL SOLVE (NK, 2, B, B(1,9), DEVN)
      C
0061        IF (GM.GT.1.4*GRAD)  GO TO 360                                    o.k.
0063             avsh = amax1(b(2,9),0.) *.5 + .5                             use mean
0064             if (b(2,9).gt.1.)  avsh = b(2,9)/avsh                        avg 1/SH
0066  340        ms = int(10.*avsh+.5)                                        10* mean
0067             devl = devn
0068             call  solve (ms, -2, b, b(1,9), devn)                         imposed
0069             if (devn.gt.devl*2.+.01 .or. avsh.eq.1.)  go to 360
0071                avsh = 1.                                                 sh= sha
0072                go to 340
0073  360   SH = SH*SQRT( ABS(B(2,9)) )
0074        PARHT = SH*AMIN1( ALOG(1.+SH*GRAD), 1.8)                          FN to FC
0075        HM = HN + PARHT
0076           FH = GIND(0., HM)
0077             if (parht.gt.sh.or.gm.lt.1.8*grad)  go to 500                 no itern
0079  400   CONTINUE
      C.....
      C-----------------------------------      Form peak parameters
0080  500   FC = EXP(B(1,9))
0081        ERR= 0.5* ( HM-SH*ALOG(1.+SH*GL) -HT(KRM-1) )                     fit last
0082        HM = HM - .9*ERR                                                    2 hts.
0083        HT(KRM)=HN + .1*ERR
0084        HT(KR)= HM
0085        PARHT = HM - HT(KRM)                                              peaksecn
0086        PARFC = FM/SQ(.5*PARHT/SH) - FM                                   FC parab
0087                     if(lbug.eq.8)print*,fc,hm,err,parht,parfc
0089             r  = 2+nk-nka
0090             fc = amax1( amin1(fc,fm+parfc*r), fm+parfc/r )                limit FC
0091        FV(KR) = FC
      c-------------------------------------------------------------------------------
      c Check overlap-- this section can be omitted with manual scalings (F monotonic)
0092        FNEXT = FV(KV+1)                                                  newlayer
0093        IF (FNEXT.GE.0.)  GO TO 620
0095           FNEXT = SQRT(FNEXT*(FNEXT+.99*FH))                               X ray
0096              DO 600  I = 2, 30
0097  600         IF (FV(KV+I).GE.0.)  GO TO 610                              1st Oray
0099  610         FNEXT = AMIN1(FNEXT, FV(KV+I))
      C
0100  620   IF (FNEXT.LE.0.)  FNEXT = FM + 4.*PARFC + .05
0102        FC = AMIN1(FC, FNEXT-.03)
0103        DFC= FV(KR) - FC                                                  reduce F
0104        FM = FM - DFC
0105        DEVN= AMIN1(DEVN*2., .5*(FNEXT-FM)/FC)
0106        IF (DFC.LE.0.01)  GO TO 680
      c                                        Layer overlap;  reduce frequencies
0108                if(lbug.ge.-9)print640, fv(kr), fc, fnext                 **----->
0110  640           format ('0* FC  reduced from',f7.3,'   to',F7.3,
     $                    '   since next layer starts at',f7.3,' MHz.' /)
0111             df = dfc/fv(kr)**3
0112                do 660  k = 1, kr
0113  660           fv(k) = fv(k) - df*fv(k)**3
0114             fa = fa - df*fa**3
0115  680   CONTINUE
      c-------------------------------------------------------------------------------
      c                                        List peak parameters
0116        DEVF = FC*DEVN                                                    freq err
0117        BAVGE= -B(2,6) - B(NKA,6)
0118        DEVS = DEVN * SH/BAVGE                                             sh err.
0119        DEVH = DEVN *PARHT/BAVGE + DEVS*SH*GRAD                            hm err.
0120        TCONT= TCONT + PARHT*(FC**2+.5*FM**2) /1.5                        parabola
0121        IF (GM.LT.1.4*GRAD)  SH = -SH                                       -model
0123           if(lbug.ge.-9)print 750, fc,devf, hm,devh,sh,devs, tcont/fc**2 **----->
0125  750      format ('0Peak', F7.3, ' (+/-',F5.3, ') MHz,    Height', F6.1,
```

```
     1           ' (+/-',F4.1,') km.    Scale Height',f5.1' (+/-',F4.1,
     2           ') km.     Slab (to peak) =',F6.1,' km.')
0126       FV(KR+4) = DEVF
0127       HT(KR+4) = DEVH
0128       FV(KV) = FC
0129       RETURN
0130       END
     c*******************************************************************************


0001                  SUBROUTINE    TRACE  ( FV,HT, TRAS )
     c*************************************************************************jan'79.
     c          Full debug listing,  produced when  0 < lbug < 10.
     c   lbug = 1 produces a single trace line for each real-height step in polan.
     c          2, 3, 4  add fuller trace outputs.
     c   lbug > 3 includes intermediate listings of freq,height.
     c   lbug = 5 adds a full list of the equations in the matrix B.
     c
0002       DIMENSION  FV(40), HT(40)
     c
0003       COMMON /POL/ B(40,17),Q(18), FH,ADIP, MODE,MOD, FA,HA, TCONT,LBUG
0004       COMMON /POL/ HS, FC,FCC, SH, PARHT, HVAL,VWIDTH,VDEPTH, XWAT
0005       COMMON /POL/ MAXB,NF, NR,NL, NX, MS,MT,JM, LK, KR,KRM, KV,MF, NC  counters
0006       DATA  LIST   / 0 /                                                 to save
     c-------------------------------------------------------------------------
0007       IF (IABS(LBUG).GT.9)  RETURN                                       no list
0009       IF (TRAS.NE.0.)  GO TO 10
0011                 PRINT 50, 'FRQ', (FV(I), I=1,15)                        ------->
0012                 PRINT 50, 'HTS', (HT(I), I=1,15)
0013                 LIST = 2                                                 head
0014                 RETURN
     c-------------------------------------------------------------------------
0015  10   IF (TRAS.EQ.2.2) PRINT 12
0017  12   FORMAT('0##-------------------------------------------------------')
     c
0018       IF (TRAS.EQ.2.2 .OR. LIST.GT.1)  PRINT 20                          ------->
0020  20   FORMAT ('0##TRACE:  kr  lk jm mt    ha     fa   frm',
     1        '  krm  kv  nf  nr  nl  nx  ms mode mod    hs',
     2        '    fc   fcc    sh   parht   hval vwidth vdepth')
     c
0021       FRM = FV(KRM)
0022       PRINT 25, TRAS, KR,LK, JM,MT, HA,FA,FRM, KRM,KV, NF,NR,NL,NX,MS,  ------->
     $                 MODE,MOD, HS, FC,FCC,SH, PARHT,HVAL,VWIDTH,VDEPTH
0023  25   FORMAT(' ##',F5.1,':',2I4,2I3, F8.2,2F6.2, 9I4, 2X,4F6.2,4F7.2)
     c
0024       LIST = 0
0025       IF (ABS(TRAS).LT.2.3.OR.LBUG.LT.4.OR.ABS(TRAS-3.3).LT..8) GO TO 60
     c
     c                           list data, at lbug > 3 and tras =2.3 OR 4.3
0027       KM = KV+NF+NX
0028       PRINT 50, 'FRQ', (FV(K), K=KV,KM)                                  ------->
0029       PRINT 50, 'VHT', (HT(K), K=KV,KM+3)
0030       PRINT 50, 'RHT', (HT(K), K=KR,KRM)
0031  50   FORMAT (' #',A3, 16F8.2/ 5X,16F8.2)
0032       LIST = 1
0033  60   IF (ABS(TRAS).NE.3.3 .OR. LBUG.NE.5)  RETURN
     c-------------------------------------------------------------------------
     c                           list matrix B,   at  lbug = 5 and tras = 3.3 only.
0035       NCOL = IABS(JM)+1
0036       PRINT 70, (J, J = 1, NCOL)                                         ------->
0037  70   FORMAT (' ### equations     J =', I4, 12I9)
     c...
0038       NLINE = NR+NF+NX+MS
0039           DO 80 LINE = 1, NLINE
0040  80       PRINT 90, LINE, (B(LINE,ICOL), ICOL=1,NCOL)                    ------->
0041  90       format (' # matrix b, line', I3, 12F9.4)
     C...
0042       LIST = 2
0043       RETURN
0044       END
     c*******************************************************************************
```

```
0001                SUBROUTINE   SOLVE  (M, N, B, Q, DEVN)
      c*********************************************************************jan77.
      c Solve m simultaneous equns in n unknowns, given in the array  B(m,n+1).
      c The result is returned in  Q(1)  to  Q(n).
      c-------------------------------------------------------------------------
      c     Adjustments to a previously obtained solution may be made by
      c             call  solve (mm, -n,  B, Q, devn).
      c  This modifies the least-squares solution by adding the constraint
      c             Q(n) = mm/10.,   with a weight of 10.
      c  Any number of the coefficients  Q  can be modified in this way.
      c
      c     When a constraint is applied to  Q(n),  any constraints
      c  previously applied at larger values of n are first removed by  solve.
      c  Previous constraints at smaller values of  n  remain.
      c     Addition of constraints is numerically stable;
      c     the removal of constraints is less accurate but still o.k.
      c-------------------------------------------------------------------------
0002        DIMENSION  B(40,17), Q(18), ISETQ(17)
0003        DATA  NQ, NP, ISETQ  / 19*0 /
      c                               (Change above  DATA  to  SAVE  for fortran 77).
0004        IF (N.LT.0)  GO TO 3
0006        NQ = N
0007        NP = N+1
      c                                  Householder transformations
0C08        K = 0
0009  1     K = K+1
0010           ISETQ(K) = 999
0011           S = 0.
0012           MK = M-K+1
0013              IF (MK.LE.0)  GO TO 7
0015           S = SUMVAL(MK,B(K,K),B(K,K),3)
0016              IF (K.EQ.NP)  GO TO 7
0018           A = B(K,K)
0019           D = SIGN(SQRT(S),A)
0020           B(K,K) = A+D
0021           C = A*D+S
0022              DO 2  J = K+1, NP
0023              S = SUMVAL(MK,B(K,K),B(K,J),3)/C
0024                 DO 2  I = K, M
0025  2              B(I,J) = B(I,J)-B(I,K)*S
0026        B(K,K) = -D
0027        GO TO 1
      c-------------------------------------------------------------------------
      c                        remove any previous constraints on q(n) to q(nq);
0028  3     NN = IABS(N)-1
0029        REMOV = -1.
0030  4        NN = NN+1
0031           IF (ISETQ(NN).NE.999)  GO TO 5
0033  51       IF (NN.LT.NQ.AND.DEVN.NE.-1.)  GO TO 4
      c                        add constraint Q(n)=m*.1 (Givens transformation).
0035        NN = IABS(N)
0036        REMOV = 1.
0037        ISETQ(NN) = M
0038  5     Q(NP) = ISETQ(NN)
0039        IF (REMOV.LT.0.)  ISETQ(NN) = 999
0041        Q(NN) = 10.
0042        DO 6  I = NN, NP
0043           BII = B(I,I)
0044           R = SQRT( AMAX1(BII**2+Q(I)**2*REMOV, .1E-9) )
0045           C = BII/R
0046           S = Q(I)/R
0047              DO 6  J = I, NP
0048              QJ = Q(J)
0049              IF (I.EQ.NN.AND.J.NE.I.AND.J.NE.NP)  QJ = 0.
0051              Q(J) = C*QJ-S*B(I,J)
0052  6        B(I,J) = C*B(I,J)+S*QJ*REMOV
0053        IF (REMOV) 51,51,8
      c-------------------------------------------------------------------------
      c                                  back substitution
0054  7     B(NP,1) = M
```

```
0055          B(NP,NP)= SQRT(S)
0056    8     DEVN = ABS(B(NP,NP))/SQRT(B(NP,1))
0057          DO 10  II = 2, NP
0058             I = NP-II+1
0059             S = B(I,NP)
0060                DO 9  J = I+1, NQ
0061    9           IF (II.GT.2)  S = S - Q(J)*B(I,J)
0063    10       Q(I) = S/B(I,I)
0064          RETURN
0065          END


0001                    FUNCTION   SUMVAL  (N, A, B, L)
      c*************************************************************************jan77.
      c  Returns the sum for  j = 1 to N  of:-
      c       a(j)*b**j  at  L = 1 ;      j*a(j)*b**(j-1)  at  L = 2  (using b = B(1)).
      c       a(j) *b(j) at  L = 3 ;      a(j)*b(1-L+j*L)  at  L > 3  (giving  LL = 4).
      c-------------------------------------------------------------------------
0002          DIMENSION  A(9), B(9)
0003          J = N
0004          S = 0.
0005          LL= MINO(L,4)
      c.....................            loop from  j = n  to  j = 1.
0006    9     GO TO (1,2,3,4), LL
      c                           L = 1;  evaluate polynomial  a(j)*b**j
0007    1     S = (A(J)+S)*B(1)
0008          GO TO 8
      c                           L = 2;  calculate gradient sum a(j)*j*b**(j-1)
0009    2     S = S*B(1)+A(J)*FLOAT(J)
0010          GO TO 8
      c                           L = 3;  calculate sum of products  a(j)*b(j)
0011    3     JB= J
0012          GO TO 6
      c                           L > 3;  sum of a(j)*b(1,j),  with L = 1st dim of b
0013    4     JB= (J-1)*L+1
0014    6     S = S+A(J)*B(JB)
      c
0015    8     J = J-1
0016          IF (J.GT.0)  GO TO 9
      c.....................
0018          SUMVAL = S
0019          RETURN
0020          END


0001                    FUNCTION   GIND  (F, T)
      c*************************************************************************jan77.
      c  Group index subroutine, for o or x rays.      (f negative for x ray).
      c  Gives (group index -1) to full machine accuracy,  for any value of t.
      c Initialise by  call gind(GH,-dip) to set gyrofreq (mhz) and dip (deg).
      c GH is ground value;  scaled to height h by  FH = gind(0.,h), with h>2.
      c       An initial negative value of GH suppresses the scaling.
      c-------------------------------------------------------------------------
0002          DATA  GH,GHSN,GCSCT, FH,FHSN,FCSCT,HFH, C,C2  /  9*0. /
      c                                          (change data to save for fortran77).
0003            IF (F.EQ.0.)  GO TO 2
0005            IF (T.LT.0.)  GO TO 1
0007            T2 = AMAX1(T,.1E-9)**2
0008            IF (F)  5,6,6
      c                               store ground constants
0009    1     GH = F
0010          FH = ABS(GH)
0011          DIP = -.01745329*T
0012          GHSN = GH*SIN(DIP)
0013          GCSCT= (GH*COS(DIP))**2*.5/GHSN
0014          GO TO 3
      c                               scale constants to height  h  =  t.
0015    2     IF (GH.LT.0. .OR.T.LT.2.)  GO TO 4
0017          FH = GH/(1.+T/6371.2)**3
0018    3     FHSN = GHSN*FH/GH
```

```
0019        FCSCT= GCSCT*FH/GH
0020        HFH = .5*FHSN
0021           C = FCSCT+FHSN
0022           C2= C*C
0023  4     GIND = FH
0024        RETURN
      c                                calculate group index  gind  =  mu' - 1.
      c                                indented sections increase accuracy for x ray.
0025  5        X  = (F+FH)*T2
0026           G1 = X-FH
0027           GO TO 7
0028  6     G1 = F*T2
0029  7     G2 = FCSCT/G1
0030        G3 = SQRT(G2*G2+1.)
0031           IF (F.GE.0.)  GO TO 8
0033           G4 = FHSN*(G3-G2)
0034           X  = X*(G1-FH)
0035           G5 = -G4*X/ (FHSN*(SQRT(X+C2)+C))
0036           GO TO 9
0037  8     G4 = FHSN/(G2+G3)
0038        G5 = G1+G4
0039  9     G6 = F+G4
0040        G7 = (F*G2*G4/G1-HFH)*(F-G1)/(G3*G6)
0041        GIND = ABS(G7+G6)/SQRT(G5*G6) -1.
0042        RETURN
0043        END
```

F.5. THE PROGRAM SCION

This mainline program gives an example of a complete real-height analysis procedure using POLAN. SCION controls the scaling of data from a projected ionogram, correction for skewed or non-linear axes; conversion of the digitised values into virtual height and frequency; ordering of separate ordinary and extraordinary ray data in the way required by POLAN; analysis of the data and listing of the calculated real heights. The digitising sections run in conjunction with a Hewlett Packard 9864A digitiser and digitising table, and will need modifying for other equipment. A full description of the operation and use of SCION is given in section 11 of this report.

```
C    --   S C I O N   --        INPUT on UNIT 6,  OUTPUT on UNIT 7.
C
C   Program to perform an N(H) analysis of an ionogram scaled on the X-Y
C   reader,  using the generalised polynomial analysis program  POLAN.
C   Leo McNamara,  August 1978;   ----   John Titheridge,  March 1981.
C================================================================================
C================================================================================!
C:-  SUMMARY SCALING INSTRUCTIONS.      [...] / [...]  denotes alternatives.    !
C:-       NULL = bottom left corner,    outside ionogram frame;                 !
C:-       TL   = Top Left corner;       TR  = Top Right corner.                 !
C--------------------------------------------------------------------------------!
C 1.  Scale O-ray data for first layer.                                         !
C 2.  [scale TL]  /  [scale FC;  scale TR].                                     !
C                                                                               !
C 3.  Scale O-ray data for next layer.                                          !
C 4.  [scale TL]  /  [scale FC;  scale TR].                                     !
C 5.                Repeat steps  3. and 4.,  until no more O data.             !
C                                                                               !
C 6.  Scale  NULL.       ( this always denotes  'end of ionogram').             !
C----                                                                           !
C 7.  Scale lowest useable X-ray data   (over a range of about 0.6 MHz);        !
C                            /  [or, in the absence of data,  scale  TL].       !
C 8.  [Scale TL]  /  [Scale FCX for that layer;  scale TR].                     !
C 9.                Repeat steps  7. and 8.,  for each available layer.         !
C                                                                               !
C 10. scale two nulls.                                                          !
C================================================================================
```

```
C=======================================================================
C                          INPUTS  [reads]  are marked and numbered by   <<<<<1
C
C                                  Normal PRINTED OUTPUTS are shown by   *****1
C
C  LOOPS are delimited by   C.....
C
C-----------------------------------------------------------------------
      DIMENSION HEADC(19), XC(4),YC(4), FM(20),HM(20)
      DIMENSION FMX(20),  FMY(20),   HMX(20), HMY(20)
      DIMENSION FOX(100), HOY(100),  FXX(50), HXY(50)
      DIMENSION FV(150),  HT(150),   FS(120), HS(120)
      EQUIVALENCE  (FOX, FV), (HOY, HT), (FXX,FV(101)), (HXY,HT(101))
C=======================================================================
900   FORMAT ('1IONOGRAM'I3,//19A4, 'DIP,  FB,  START =' 5F7.2 )
901   FORMAT (I4,19A4)
902   FORMAT (20F4.0)
904   FORMAT (12F5.0)
907   FORMAT ('0*',6F6.2,4X,6F6.2,4X,6F6.2,4X,2F6.2)
908   FORMAT (12F5.2)
909   FORMAT (/1H0,A5,12F10.3/(6X,12F10.3))
910       FORMAT (1H0,I4,1X, A7,' POINTS DIGITIZED AND DESKEWED'/)
912       FORMAT ( 6( F7.3,F8.2,5X) )
990   FORMAT(//9X'>>>>> DO YOU WISH TO SAVE  TAPE7  FOR PLOTTING <<<<<')
      NDIM = 150
      ION  = 0
C//////////////////////////////////////////////////////////////////////
C
C                    (A). START OF NEW IONOGRAM:-
C (1). Read comments
C
10    READ (6,901) IEXIT, HEADC                                        <<<<<1
          IF (IEXIT.EQ.0)  PRINT 990
          IF (IEXIT.EQ.0)  STOP '*ALL DONE*'
      ION = ION + 1
C-----------------------------------------------------------------------
C (2). Read dip angle, gyrofrequency,  default starting height at 0.5 MHz.
C      This default height is used if no suitable X-ray data are provided.
C
      READ (6,904) DIP, FB, START                                      <<<<<2
      PRINT 900,  ION,  HEADC,  DIP, FB, START                         *****1
C
C (3). Read in the values of the visible frequency markers, to be scaled.
C                  count the frequency markers,  and list.
C
      READ (6,902) (FM(I),I = 1,20)                                    <<<<<3
          DO 20 I = 1,20
          IF (FM(I).EQ.0.0)   GO TO 22
22    NFM = I - 1
          PRINT 24 , (FM(I), I = 1, NFM)                               *****2
24        FORMAT ('0FREQ. MARKERS',20F6.1)
C
C (4). Read in the values of the height markers to be scaled.
C
      READ (6,902)   (HM(I), I = 1,20)                                 <<<<<4
          DO 30 I = 2, 20
30        IF (HM(I).EQ.0.)   GO TO 35
35    NHM = I - 1
          PRINT 40, (HM(I), I = 1, NHM)                                *****3
40        FORMAT ('0HEIGHT MARKRS',20F6.1)
C//////////////////////////////////////////////////////////////////////
C
C                                B E G I N   S C A L I N G
C
C             (B). READ THE POSITIONS OF THE FOUR CORNERS OF THE IONOGRAM
C-------                              (X,Y;  clockwise from bottom left).
C
      READ (6,908)  (XC(I), YC(I),  I = 1,4)                           <<<<<5
C
C-------   Read in the frequency marker positions - max 6 points per line.
```

```
      READ (6,908)  (FMX(I),FMY(I), I = 1,NFM),  END                    <<<<<6
C
C-------   Read in the height marker positions.
      READ (6,908)  (HMX(I),HMY(I), I = 1,NHM),  END                    <<<<<7
C                                          Deskew corners and markers
        XC1 = XC(1)
        YC1 = YC(1)
      CALL  DESKEW  ( 0, XC, YC)
      CALL  DESKEW  (NFM, FMX, FMY)
      CALL  DESKEW  (NHM, HMX, HMY)
C//////////////////////////////////////////////////////////////////////////
C
C               (C). READ IN THE O RAY VIRTUAL HEIGHTS AND FREQUENCIES
C
C               Terminate each layer at corner  2  if  FC  is  not scaled,
C                                or at corner  3  if  FC  is  scaled.
      N1 = 1
C.....
100       N6 = N1+5
          READ (6,908)  (FOX(I),HOY(I), I = N1,N6)                      <<<<<8
          N1 = N1+6
          IF (FOX(N6).GT.XC1.OR.HOY(N6).GT.YC1)  GO TO 100
C.....
      CALL  DESKEW (N6, FOX, HOY)
C
C                                Count the number of o-ray points
C.....                           Set zeros at layer terminations.
      DO 150  I = 1, N6
      F = FOX(I)
      H = HOY(I)
          IF (F.LT.XC(1).AND.H.LT.YC(1))  GO TO 200
          IF (F.LT.XC(2).AND.H.GT.YC(2))  FOX(I) = 0.
          IF (F.GT.XC(3).AND.H.GT.YC(3))  HOY(I-1) = 0.
150   CONTINUE
C.....
C                                END of all O-ray data; add another zero.
200   FOX(I) = 0.0
      HOY(I) = 0.0
      I = I + 1
      FOX(I) = 0.0
      HOY(I) = 0.0
      NOP = I
C                                LIST scaled and deskewed O-ray data
          PRINT 910, NOP,  'O-RAY'                                      *****4
C
          PRINT 912, (FOX(I), HOY(I),  I = 1, NOP)                      *****5
C//////////////////////////////////////////////////////////////////////////
C
C                   (D). READ IN THE X-RAY VIRTUAL HEIGHT DATA.
C
C                    At least one line is necessary here (off-scale point)
      N1 = 1
C.....
300       N6 = N1+5
          READ (6,908)  (FXX(I), HXY(I), I = N1,N6)                     <<<<<9
          N1 = N1+6
          IF (FXX(N6).GT.XC1.OR.HXY(N6).GT.YC1)   GO TO 300
C.....
      CALL  DESKEW (N6, FXX, HXY)
C
C                                Count the number of X-ray points
C.....                           Set zeros at layer terminations.
      DO 350  I = 1, N6
      F = FXX(I)
      H = HXY(I)
          IF (F.LT.XC(1).AND.H.LT.YC(1))  GO TO 380
          IF (F.LT.XC(2).AND.H.GT.YC(2))  FXX(I) = 0.
          IF (F.GT.XC(3).AND.H.GT.YC(3))  HXY(I-1) = 0.
350   CONTINUE
C.....
```

```
C                                   END of all X-ray data; add another zero.
380   FXX(I) = 0.0
      HXY(I) = 0.0
      NXP = I
C                                   LIST scaled and deskewed X-ray data.
          PRINT 910, NXP, 'X-RAY'                                          *****6
          PRINT 912, (FXX(I), HXY(I),  I = 1, NXP)                         *****7
          PRINT 910
C/////////////////////////////////////////////////////////////////////////
C
C                (E). ALL DATA HAS BEEN READ IN AND DESKEWED.
C             Convert the O and X ray positions to frequencies and heights
C             using 4-point interpoln in marker positions. Preserve zeros.
C
C                                         frequencies  (log interpolation)
      CALL  INTERP  (-NFM, FMX, FM )
      CALL  INTERP  ( NOP, FOX, FOX)
      CALL  INTERP  ( NXP, FXX, FXX)
C                                         heights
      CALL  INTERP  (-NHM, HMY, HM )
      CALL  INTERP  ( NOP, HOY, HOY)
      CALL  INTERP  ( NXP, HXY, HXY)
C
              PRINT 410
410           FORMAT ('0     CONVERTED TO FREQUENCY AND HEIGHT.'/)       *****8
              PRINT 912, (FOX(I),HOY(I), I=1,NOP)                        *****8
              PRINT 912
              PRINT 912, (FXX(I),HXY(I), I=1,NXP)                        *****9
C/////////////////////////////////////////////////////////////////////////
C
C            (F). ARRANGE THE SCALED POINTS IN THE ORDER REQUIRED BY POLAN.
C Array positions are  NM in combined data,  NOV in O-data,  NXV in X-data.
      NM  = 1
      NOV = 1
      NXV = 1
C---------------------            STORE X-ray points,  less final null
500   FCX = 0.
      IF (NXV.GE.NXP)   GO TO 550
C.....
          DO 520  I = NXV, NXP
              FS(NM) = -FXX(I)
              HS(NM) =  HXY(I)
            IF (FXX(I).EQ.0.)  GO TO 540
            IF (HXY(I).EQ.0.)  GO TO 530
520       NM = NM + 1
C.....
C                                   Save X-ray critical frequency
530       FCX = FXX(I)
540   NXV = I + 2
C
C---------------------            STORE O-ray points;  retain null at FC
C.....
550       DO 560  I = NOV, NOP
              FS(NM) = FOX(I)
              HS(NM) = HOY(I)
            NM = NM + 1
          IF (FOX(I).EQ.0.)  GO TO 580
560       IF (HOY(I).EQ.0.)  GO TO 570
C.....
C                                             Delete null after scaled FC
570   I = I + 1
580   NOV = I + 1
      HS(NM-1) = 0.
      IF (FCX-.5*FB.LT.FS(NM-2))  GO TO 590
          FS(NM) = FCX
          HS(NM) = 0.
          NM = NM + 1
590   IF (HOY(NOV).NE.0.0)  GO TO 500
C---------------------
C                                   ----  END DATA  ----
```

```
      FS(NM) = 0.0
      HS(NM) = 0.0
C////////////////////////////////////////////////////////////////////////
C
C                       (G).   L I S T    AND    A N A L Y Z E
C
      PRINT 909, 'FREQ ', (FS(I), I = 1, NM)                          ****10
      PRINT 909, 'HVIRT', (HS(I), I = 1, NM)                          ****10
      PRINT 910
C
C                     Write out the data to UNIT7 for plotting purposes
          WRITE (7,980)  NM, HEADC
          WRITE (7,981) (FS(I),HS(I),I = 1,NM)
C.....
          DO 940  I = 1, NM
          FV(I) = FS(I)
940       HT(I) = HS(I)
C.....
      N = NDIM
      AMODE = 0.0
C--------------------
      CALL POLAN (N, FV,HT, FB, DIP, START, AMODE,  0., 0)
C
      PRINT 909, 'FREQ ', (FV(I), I = 1, N)                           ****11
      PRINT 909, 'HREAL', (HT(I), I = 1, N)                           ****11
C
          WRITE  (7,980) N
          WRITE  (7,981) (FV(I), HT(I), I = 1, N)
980       FORMAT (I4,19A4)
981       FORMAT (6(F6.2,F6.1))
C
C                               RETURN FOR NEXT IONOGRAM
      GO TO 10
      END



            SUBROUTINE  DESKEW  (N, X, Y)
C
C         To deskew coordinates in an arbitrary quadrilateral frame.
C-------------------------------------------------------------------------
C(A).               -:*  I N I T I A L I Z E  *:-
C AT N = 0:-   X(1),Y(1)  to  X(4),Y(4)   give the coordinates of the four
C           corners of the quadrilateral, clockwise from the bottom left.
C-------------------------------------------------------------------------
C(B).               -:*  D E S K E W  *:-
C AT N greater than 0:-        Deskew  N  coordinates in the arrays  X, Y.
C=========================================================================
          DIMENSION  X(9), Y(9),  Z(4)
          DATA  X1,X2,X3, Y1,Y2,Y3  /  6 * 0.0  /
      NN = N
      IF (N.GT.0)  GO TO 60
C---------------------------------------
C(A).                           N = 0 ;    calculate the deskew coefficients
      X1 = X(1)
      Y1 = Y(1)
      X2 = (Y(4)-Y1) / (X(4)-X1)
C
      DO 20 I = 2, 4
20        Z(I) = Y(I) - Y1 - X2 *(X(I) - X1)
      Y2 = (X(2) - X1) / Z(2)
      Z3 = X(3) - X1 - Y2 * Z(3)
      Z4 = X(4) - X1 - Y2 * Z(4)
      X3 = (1. - Z(2)/Z(3)) / Z3
      Y3 = (Z3 - Z4) /(Z(2) * Z3)
C
      NN = 4
C---------------------------------------
C(B).                           N > 0 ;    deskew  the  arrays   X(N), Y(N).
```

```
C.....
60    DO 80  I = 1, NN
          B = Y(I) - Y1 - X2 * (X(I) - X1)
          A = X(I) - X1 - Y2 * B
          Y(I) = B - X3 * A * B
80    X(I) = A - Y3 * A * Y(I)
C.....
      RETURN
      END




          SUBROUTINE  INTERP  (M, X, Y)
C
C  To obtain true values of the variable Y,  by interpolating the
C  corresponding scaled ordinate  X  in a previously given marker array.
C     ****  Uses  4- term polynomial interpolation.
C     ****  Requires a minimum of 2, but preferably 4, marker points  ****
C     ****  Zeros are not altered.
C-----------------------------------------------------------------------
C USE:-
C AT M NEGATIVE,  X   is an array of  N  scaled ordinates,   corresponding
C                 to the  N  reference marker values given in Y(1) to Y(N)
C------
C AT M POSITIVE,  X(1)  to  X(N)  give  scaled  ordinates,
C                 Y(1)  to  Y(N)  return the corresponding interpd values.
C=======================================================================
C  The array  D(99)  stores the marker position (X),  its value (Y),
C                           and corrected differences for each interval.
      DIMENSION  X(20), Y(20), D(99)
C
      DATA D,  NM1, NM2, LOG  /  99* 0.0,  3* 0 /
C
      N = IABS(M)
      IF (M.GT.0)  GO TO 100
          NM1 = N - 1
          NM2 = N - 2
          LOG = 0
          IF (Y(NM1).LT.100. .AND. Y(NM1).GT.0.)  LOG = 1
C=======================================================================
C(A).     M  NEGATIVE:-        Store marker data and interpolation table.
C
      JS =  N + NM1
      KS = JS + NM2
      LS = KS + NM2 - 1
      D(KS+3) = 0.                                                  FOR M =2

      D(LS+4) = 0.                                                  FOR M =3

C.....
      DO 50  I = 1, N
          D(I) = X(I)
          D(N+I) = Y(I)
          IF (LOG.EQ.1)  D(N+I) = ALOG(Y(I))
      IF (I.LT.2)  GO TO 50
              J = JS + I
              D(J) = (D(N+I) - D(NM1+I)) /(D(I) - D(I-1))
      IF (I.LT.3)  GO TO 50
              K = KS + I
              D(K) = (D(J) - D(J-1)) / (D(I) - D(I-2))
      IF (I.LT.4)  GO TO 50
              L = LS + I
              D(L) = (D(K) - D(K-1)) / (D(I) - D(I-3))
50    CONTINUE
C.....
      RETURN
C=======================================================================
C(B).   M  POSITIVE:-          Interpolate  X(I)  in the stored array D(I)
```

```
C                  The result is returned in  Y(I) [which may equal X(I)]
C.....
100   DO 250  J = 1, N
      AX = X(J)
              YY = AX
              IF (AX.EQ.0.0)  GO TO 250
C                                          find interpolation segment INT
      INT = 2
      DO 150  I = 2, NM2
150       IF (AX.GT.D(I))  INT = I
C                                          calculate  Y(J)
      IMT = INT + 2                                                  TOP XVAL

      NS = NM1 *4 + INT
        YY = D(NS)
      NM3= NM1 - 2
C...
      DO 200  I = 1, 3
          NS = NS - NM3 - I
200       YY = YY* (AX - D(IMT-I) ) + D(NS)
C...
      IF (LOG.EQ.1)  YY = EXP(YY)
250   Y(J) = YY
C.....
      RETURN
      END
```