A DESCRIPTION OF PREDICTION ERRORS ASSOCIATED WITH THE T-BUS-4 NAVIGATION MESSAGE AND A CORRECTIVE PROCEDURE

Frederick W. Nagle

NOAA/NESDIS Satellite Applications Laboratory
Systems Design and Applications Branch

1. INTRODUCTION

During his visit to the United States in 1985 spent at CIMSS, Madison, Wisconsin, Dr. Brian Taylor of the New Zealand Meteorological Service, complained of navigational errors in the predicted position of the NOAA-9 satellite when using orbital parameters obtained from Part IV of the T-Bus message (sample attached) provided to international users of NOAA data. The existence of the errors was confirmed by other members of the International TOVS Study Conference (ITSC) held at Igls, Austria during February 1985; an action item was recommended to ascertain the nature of the problem, and to report possible remedial action at the next ITSC meeting to be held in Madison, Wisconsin in the summer of 1986. This report addresses the investigation accomplished at CIMSS.

2. DESCRIPTION OF PROBLEM

For this investigation, orbital parameters obtained from three sources were used. The first set was obtained from NOAA/NESDIS, Orbital Mechanics Branch, Suitland, Maryland, using the subroutine PSCEAR installed on the NAS 9000 system in Suitland. Definitive (not predicted) parameters were gathered daily over a period of several weeks from December 1985 through February 1986. Satellite positions obtained from the most recent PSCEAR parameters were considered the best available approximation to the truth, and were the standard against which the predicted positions were compared. The definitive parameters were usually used within one day of their epoch, and never more than three. A second set of parameters was obtained approximately daily from the T-Bus Part IV message received on the McIDAS system at Madison. This is the same set available to Dr. Taylor and others.

The T-Bus orbital parameters have the somewhat unusual property that their epoch is the time of an ascending Equator crossing. Presumably, a number of users around the world use a satellite prediction system predicated on the assumption that the parameters are valid at the instant of an Equator crossing. However, this involves a possibly suspect interpolation since primary navigation parameters are not provided in this way, and this question has been considered by generating a third set of "pseudo" T-Bus parameters at SDAB, Madison, for comparison with the T-Bus parameters cited above. These pseudo-parameters were obtained from a program which extrapolates a set of definitive parameters (our "truth") forward to the next ascending Equator crossing. The short program which performs this extrapolation is shown in the appendix below.

The basic element of the study is the prediction of satellite positions from either the actual or pseudo T-Bus, and the comparison of this prediction with the position obtained from definitive parameters. The subroutine used to predict satellite positions from a given set of orbital parameters was obtained from OMB/NESDIS, in order to use standardized software. The subroutine is known as BROLYD on the NAS 9000, and appears in the present study as the vector-valued function VBLMOD (Vector Brouwer-Lyddane Model). A complete listing of the routine on the SSEC McIDAS IBM 4381 is given below.

Figure 1 illustrates the problem of which Dr. Taylor has complained. The y axis is the epoch of whatever set of parameters is used to make the prediction. The abscissa represents the time of a prediction after the epoch of the prediction, each tick mark delineating one day, the entire range being 27 days. The ordinate represents the along-track error of the prediction with the horizontal lines positioned at ±100 kilometers. The strings of semi-continuous dots depict the navigation error as the T-Bus ages, and the continuous line is a quadratic fit to these errors. Several points can be noted. Firstly, there is an overall tendency toward degradation in that the predicted positions generally fall further and further behind with time. Secondly, the degradation occurs at a rate which is by no means uniform, for certain sets of parameters tend to produce strings of predicted positions which degrade much more rapidly than other strings produced by other sets. Thirdly, the prediction error for any given epoch is not continuous. The discontinuities are obvious and occasionally very large as evidenced by the scattering of individual dots. Finally, the error is frequently non-trivial even at the time of epoch (the origin). It should be recalled that this navigation is being used to locate the AVHRR data where an error of even a few kilometers is objectionable.

Figure 2 is similar to Figure 1, but is derived from the pseudo T-Bus parameters produced in Madison. In some ways, there is an improvement. The short term error is reduced and it appears that there is somewhat less long-term bias. However, the scatter of the predicted positions is worse, with some sets of parameters producing positive, and others negative errors, which become rapidly worse with time. Hence, for long-term predictions, neither set is acceptable.

3. A METHOD OF REDUCING THE ERROR

We have not been successful in locating the source of the error in the predictions from the T-Bus. Its elimination appears to involve an in-depth study of the orbital prediction model which is clearly beyond the scope of our effort. However, we can offer a palliative which will generally make the navigation suitable over periods of up to a week.

The fact that large along-track errors occur using either set of T-Bus parameters suggests that for whatever reason, the <u>mean</u> orbital period is badly predicted. Yet, the true orbital period is rather well-known, for it can be obtained from the known times of satellite equator crossings found in either the true or pseudo T-Bus parameters.

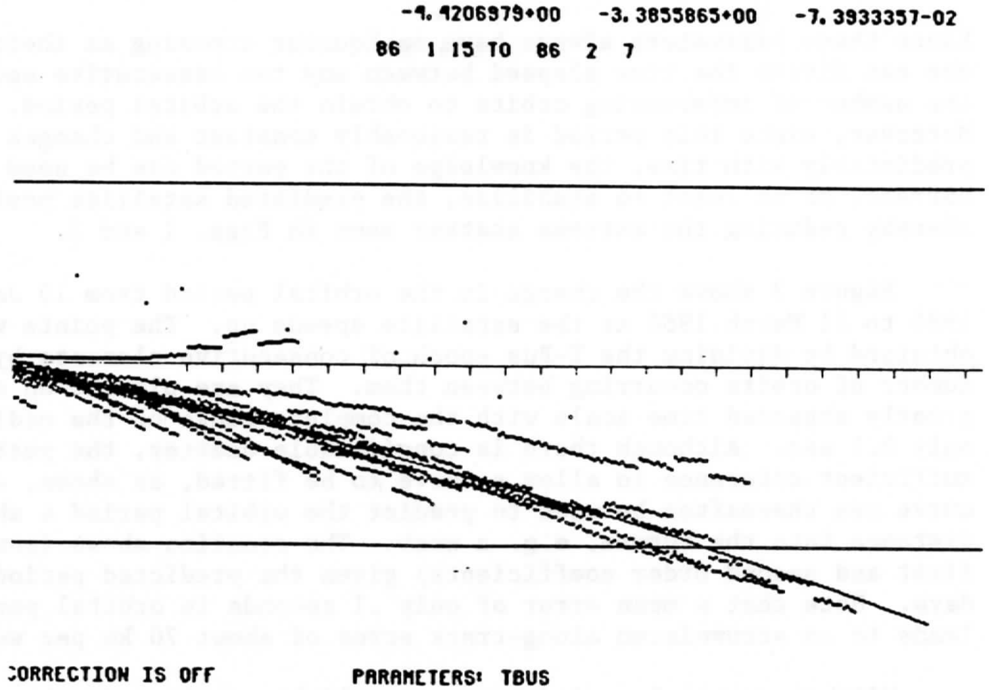

Figure 1. Prediction errors from standard T-Bus parameters; no periodic stabilization.

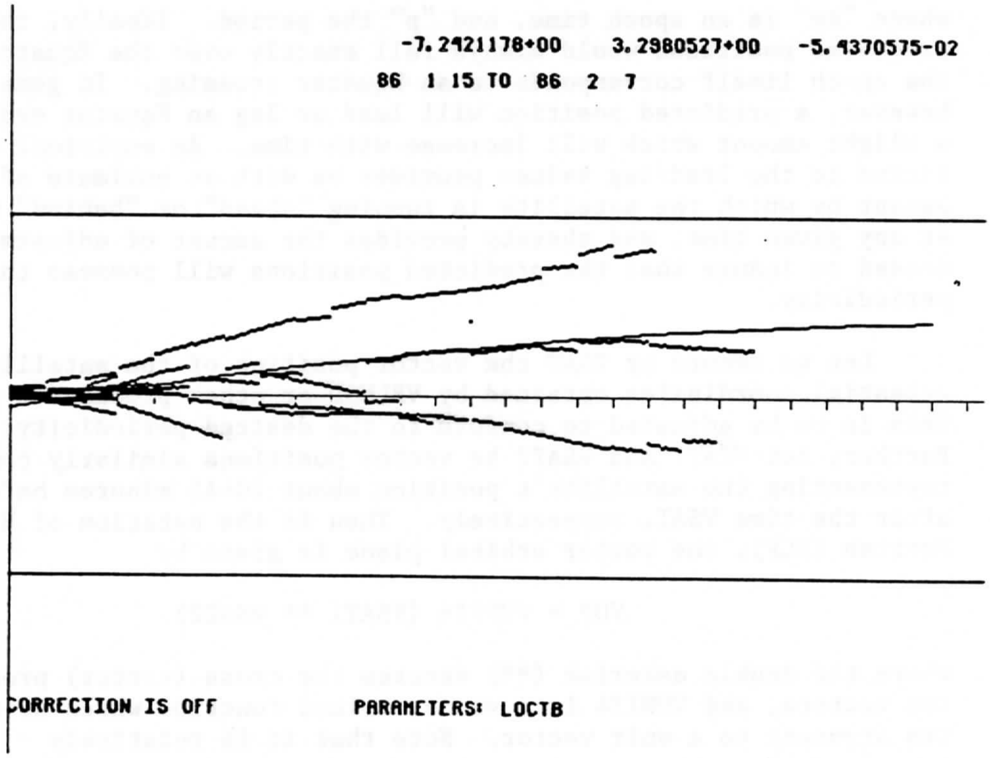

Figure 2. Prediction errors from CIMSS-generated pseudo-T-Bus parameters; no periodic stabilization.

Since these parameters always have an Equator crossing as their epoch, one can divide the time elapsed between any two consecutive epochs by the number of intervening orbits to obtain the orbital period. Moreover, since this period is reasonably constant and changes predictably with time, the knowledge of the period can be used to correct, or at least to stabilize, the predicted satellite positions, thereby reducing the extreme scatter seen in Figs. 1 and 2.

Figure 3 shows the change in the orbital period from 15 January 1986 to 11 March 1986 as the satellite speeds up. The points were obtained by dividing the T-Bus epoch of consecutive elements by the number of orbits occurring between them. They are plotted on a greatly expanded time scale with the complete range of the ordinate only 0.1 sec. Although there is considerable scatter, the pattern has sufficient coherence to allow a curve to be fitted, as shown, and this curve can thereafter be used to predict the orbital period a short distance into the future, e.g. a week. The equation shown (constant, first and second order coefficients) gives the predicted period in days. Note that a mean error of only .1 seconds in orbital period leads to an accumulated along-track error of about 70 km per week.

With the orbital period known or reliably estimated, the predicted satellite positions can be much stabilized as follows. Beginning at the epoch of a given set of parameters, satellite positions are predicted at every tenth orbital period thereafter, at the times

ep, ep + 10*p, ep + 20*p, ep + 30p, etc.

where "ep" is an epoch time, and "p" the period. Ideally, the predicted positions would always fall exactly over the Equator, since the epoch itself corresponds to an Equator crossing. In general, however, a predicted position will lead or lag an Equator crossing by a slight amount which will increase with time. An empirical curve fitted to the lead/lag values provides us with an estimate of the amount by which the satellite is running "ahead" or "behind" schedule at any given time, and thereby provides the amount of adjustment needed to insure that the predicted positions will possess the desired periodicity.

Let us denote by VSAT the vector position of the satellite in celestial coordinates obtained by VBLMOD or other prediction program. This is to be adjusted to conform to the desired periodicity. Further, let VSAT1 and VSAT2 be vector positions similarly computed representing the satellite's position about 10-15 minutes before and after the time VSAT, respectively. Then in the notation of High-Level Fortran (HLF), the vector orbital plane is given by

VOP = VUNIT4 (VSAT1 ** VSAT2)

where the double asterisk (**) denotes the cross (vector) product of two vectors, and VUNIT4 is a vector-valued function which normalizes its argument to a unit vector. Note that it is relatively

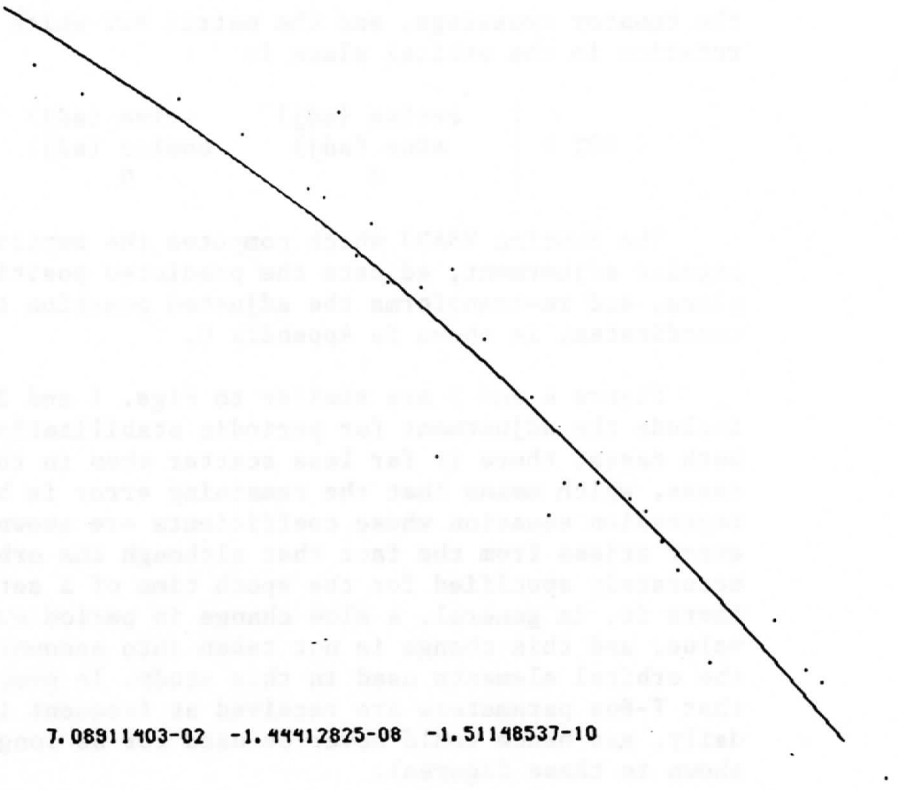

Figure 3. Change in NOAA-7 orbital period 15 January 1986 to 11 March 1986. The entire range is only about .1 second.

unimportant if the two positions VSAT1 and VSAT2 contain slight along-track errors, since the vector orbital plane precesses only very slowly, about one-fourteenth of a degree per orbit.

The vector orbital plane VOP is one of three orthonormal vectors which constitute a dextral coordinate set attached to the plane of the orbit. VOP is normal to the plane of the orbit, and two other orthonormal vectors in this plane can easily be found, as follows. Let VEQ by the projection of VOP onto the plane of the Equator. Then the cross-product VX = VEQ**VOP is a vector lying along the intersection of the orbital and equatorial planes. Finally, VY = VOP**VX points from the center of the earth, in the orbital plane, toward the point of maximum satellite latitude. For example,

```
VEQ = VEC4 (vop(1), vop(2), 0.)
VX  = VUNIT4 (VEQ ** VOP)
VY  = VOP ** VX
```

The vector-valued function VEC4 returns a vector whose three components are its three arguments. The three unit vectors thus found (VOP, VX, VY) may themselves be regarded as the columns of a 3x3 orthogonal matrix MXR which transforms an arbitrary vector from the basis of the orbital plane to the celestial basis. Moreover, its inverse is also its transpose, and transforms a vector from the celestial to the orbital basis. The angular adjustment "adj" to be made to the satellite's position is known from the lead/lag values in the Equator crossings, and the matrix ROT which accomplishes this rotation in the orbital plane is

$$\mathrm{ROT} = \begin{vmatrix} \text{cosine (adj)} & \text{-sine (adj)} & 0 \\ \text{sine (adj)} & \text{cosine (adj)} & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

The routine VSADJ which computes the empirical equation for angular adjustment, adjusts the predicted position in the orbital plane, and re-transforms the adjusted position to celestial coordinates, is shown in Appendix C.

Figure 4 and 5 are similar to Figs. 1 and 2, respectively, but include the adjustment for periodic stabilization described above. In both cases, there is far less scatter than in the two unstabilized cases, which means that the remaining error is better described by a regression equation whose coefficients are shown. This remaining error arises from the fact that although the orbital period can be accurately specified for the epoch time of a set of orbital elements, there is, in general, a slow change in period even from this estimated value, and this change is not taken into account over the lifetime of the orbital elements used in this study. In practice, it is assumed that T-Bus parameters are received at frequent intervals, perhaps daily, and hence would never be used for as long as three weeks (as shown in these figures).

4. CONCLUSIONS AND RECOMMENDATIONS

It appears feasible even with imperfect orbital elements and imperfect prediction programs to upgrade significantly the quality of NOAA-9 (or other) satellite positions by making use of the orbital period, which can be reliably estimated. To achieve this end, the following general approach can be used, although its precise application by any user will depend on the resources available at his site.

(a) An archive of recent T-Bus orbital elements must be available. The fact that their epoch coincides with an Equator crossing is desirable, because Equator-crossing is a moment of the orbit at which orbital periods can be easily measured. If no T-Bus elements are available, but definitive elements are known, the former can be obtained from the latter using, for example, a routine such as shown in Appendix A.

(b) With T-Bus or pseudo T-Bus elements available, a program like that in Appendix B can be used to obtain a regression equation for predicting orbital periods in the near future.

(c) With an accurate estimate of period now available for any orbit, the known periodicity can be applied to stabilize the orbital position in order to remove both bias and scatter from the predicted positions. A routine like that shown in Appendix C can be used.

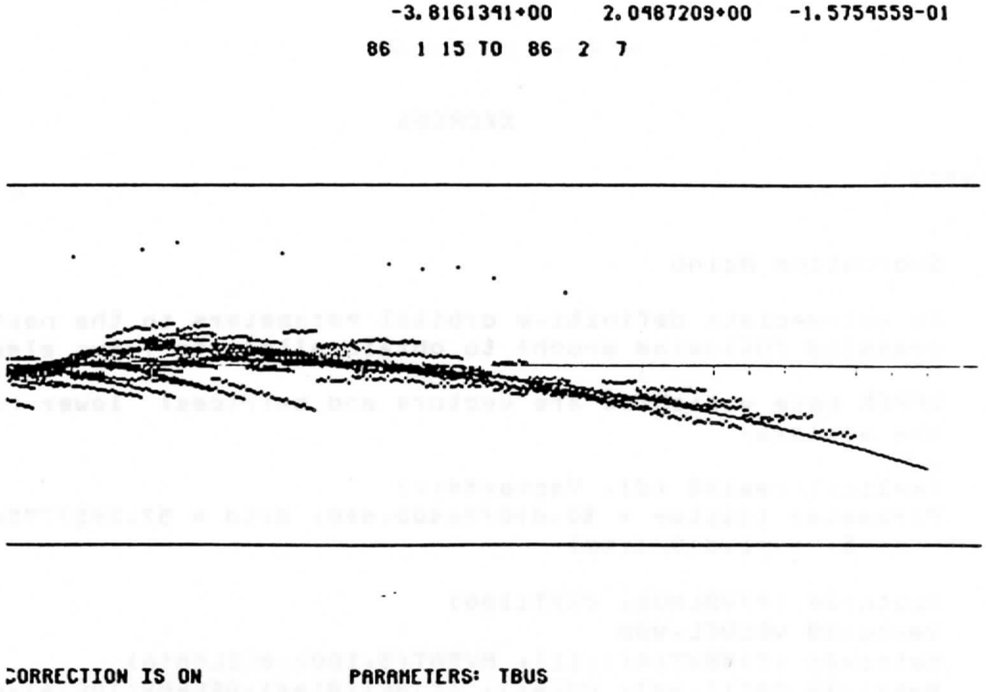

Figure 4. Prediction errors from standard T-Bus parameters using periodic stabilization.

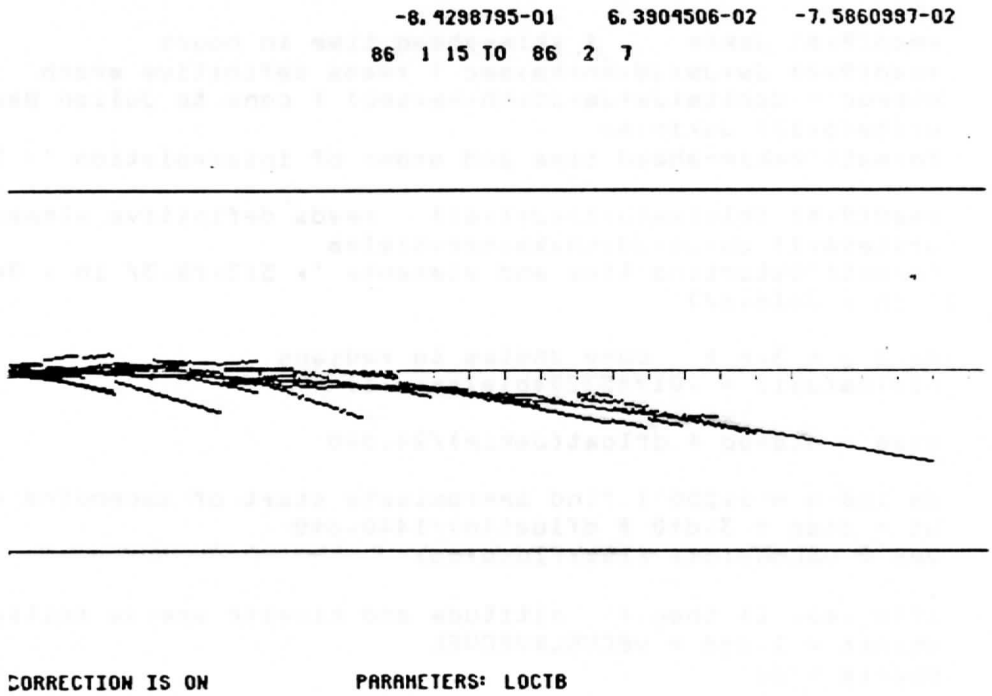

Figure 5. Prediction erros from pseudo-T-Bus parameters using periodic stabilization.

APPENDIX A

```
*
      Subroutine Main0
*
*     To extrapolate definitive orbital parameters to the next Equator
*     crossing following epoch; to obtain alternate T-Bus elements.
*
*     UPPER case variables are vectors and matrices;  lower case
*     are scalars.
*
      Implicit real*8 (d), Vector*4(V)
      Parameter (dtstep = 30.d+0/86400.d+0, drtd = 57.2957795d+0,
     1  mr=5, rx=1.d+0/drtd)
c
      Vector*4 (f)VBLMOD, VSAT(100)
      Vector*8 VECVEL,VBW
      Matrix*4 (f)MSUBI4(1,11), MVSAT(3,100),BLELEM(6)
      Matrix*8 AA(11,mr), VV(mr), (f)MFIT8(mr),DELEMS(100,6),UPDELS(6),
     1  MX(11),MCOEFS(mr,1),(f)MSUBI8(11,6),MSUB(11,6),
     2  DTT(100),(f)MALL8(100),DINELS(6)
*
      common/blxtra/dtprep,UPDELS,VECVEL ; Updated BL elements
      common/vbl/ dtepoc,BLELEM
*
      equivalence( VSAT(1), MVSAT)
*
c     cccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      read(9,*) jskip      ; skip-ahead time in hours
      read(9,*) jy,jm,jd,kh,km,sec ; reads definitive epoch
      dtepoc = dabtim(jy,jm,jd,kh,km,sec) ; conv to Julian Day Number
      write(6,17) jskip,mr
17    format('0skip-ahead time and order of interpolation ', 2i6/)
*
      read(9,*) (blelem(j,1),j=1,6);   reads definitive elements
      write(6,1) jy,jm,jd,kh,km,sec,blelem
1     format('0starting time and elements ', 5i3,f8.3/ 1h , 3e16.6/
     1  1h , 3e16.6/)
c
      do 2 j = 3,6 ;   conv angles to radians
2     blelem(j,1) = .01745329*blelem(j,1)
c
      dtgo = dtepoc + dfloat(jskip)/24.d+0
c
      do 100 n = 1,200 ; find approximate start of ascending pass
      dt = dtgo + 3.d+0 * dfloat(n)/1440.d+0
      VWB = VBLMOD(dt, flat,flong,cd)
*
      if(n .eq. 1) then ;   altitude and kinetic energy initially
      dkinet = 1.d+6 * VECVEL*VECVEL
      dzorig = cd
```

```
        dgfa = 9.8d+0 * (6371.d+0 / dzorig)**2
        write(6,91) cd,dkinet
91      format('0Initial altitude and energy ', f9.1,2x,d18.7/)
*
        else
        dk = 1.d+6 * VECVEL*VECVEL
        dp = 1000.d+0 *dgfa * (dble(cd) - dzorig)
        dtot= dp + dk
        write(6,93) n,dk,dp,dtot
93      format(4x, i6, 3d18.8)
        end if
*
        if(flat .lt. -75.) go to 102 ;   close to mimumum latitude
100     continue
c
102     do 110 n = 1,100;  search northward in small steps
        dtt(n,1) = dt + dtstep * dfloat(n)
        VSAT(n) = VBLMOD(dtt(n,1), flat,flong,cd)
c       VBLMOD also returns updated BL elements thru common/blxtra/
110     DELEMS(n,$) = UPDELS
c
        lit = litsch(MVSAT(3,$), 100) ; search for smallest z-component
*       Litsch searches an array for the smallest absolute value.
*       The 3rd row of MVSAT is the z-components of position vectors.
*
        jgo = lit - 5;  choose 5 points on either side near Equator
        jstop = lit + 5
        MX = MSUBI4( MVSAT,3,100, 3,3,1, jgo,jstop,1)
        MCOEFS = MFIT8(MX, DTT(jgo,1), 11, mr,   VV, AA)
*
*       We just obtained mr coefficients fitting time as a function
*       of z-component near Equator crossing.  The constant term
*       (the approximation when z=0) is the crossing time.
c
        dtx = mcoefs(1,1) ;     interpolated crossing time
        call tinver( dtx,  ny,nm,nd,kh,km,ks, nth)
*       Tinver converts Julian Day Number back to civil date/time.
        write(6,11) ny,nm,nd,kh,km,ks
11      format('0approximate crossing time ', 6i3/)
c
c       Extract the sub-matrix of orbital parameters straddling Equator
        MSUB = MSUBI8( DELEMS,100,6, jgo,jstop,1, 1,6,1) ; 11x6
        DTT = DTT - MALL8( dtx, 100,1);  subtract dtx from all times
*       DTT times are now relative to Equator crossing.
c
        do 150 j = 1,6   ;  interpolate each of 6 elements to dtx
        MCOEFS = MFIT8(DTT(jgo,1),MSUB($,j),11,mr, VV,AA)
150     dinels(j,1) = mcoefs(1,1)
*
c       convert parameters to McIdas storage format
        ecc = dinels(2,1)
        ap = drtd*dinels(5,1)
```

```
      ra = drtd*dinels(4,1)
      fincl = drtd*dinels(3,1)
      sma = dinels(1,1)
      fma = drtd*dinels(6,1)
*
      call tinver(dtx, jy,jm,jd,kh,km,ks, nth)
      frac = 86400.d+0 * (dtx - dabtim(jy,jm,jd,kh,km,ks))
      sec = float(ks) + frac
      write(6,77) jy,jm,jd,kh,km,sec, sma,ecc,fincl,ra,ap,fma
 77   format(1h ,5i3,f6.2, f8.2, f9.6, f7.3, 3f8.3/)
*
*     Compute energies using the definitive elements, and also using
*     the pseudo-TBUS elements just computed.
*
      VX = VBLMOD(dtx, flat,flong,cd)
      dkin = 1.d+6 * VECVEL*VECVEL
      dpot = 1000.d+0 * dgfa * (dble(cd) - dzorig)
      dtot = dkin + dpot
      write(6,79) flat,flong,dkin,dpot,dtot
 79   format('0definitive lat/long at epoch, kinetic, potential, total'/
     1  1h , 2f9.2/1h , 3d18.8)
*
      dtepoc = dtx
      BLELEM = DINELS   ;  insert pseudo-elements just computed.
      VSUDO = VBLMOD(dtx, flat,flong,cd)
      dkin = 1.d+6 * VECVEL*VECVEL
      dpot = 1000.d+0 * dgfa * (dble(cd) - dzorig)
      dtot = dkin + dpot
      write(6,83) flat,flong,dkin,dpot,dtot
 83   format('0using pseudo-tbus parameters ', 2f9.1/1h , 3d18.7/)
*
      call mfstak(0)
      return
      end
c
*********************************************************************
      Matrix Function MFIT8*8 (x,y,n,m,  v,aa)
c
c     UPPER case symbols are matrices; lower case are scalars.
c     To obtain the m coefficients (m,1) for a least-squares fit to
c     n x,y pairs of data.  See p. 278 on 'Approximation' in Froberg.
C     The functional value returned by this routine is the (m,1)
c     matrix of real*8 fitting coefficients, in ascending powers.
C     If this routine is called from a conventional Fortran program,
c     the call is...
C
c     This routine is similar to MATFIT, but has REAL*8 inputs.
c
C     CALL MFIT8( XLIST, YLIST, NPAIRS, MORDER,  V,AA,  COEFS)
C                   8      8      I4      I4     8  8     8
c
      Matrix*8 MFIT8(m),AA(n,m), (f)MTRAN8(m,n), (f)MSYMVT(m,m),
```

```
       1  V(m),X(n),Y(n)
c
          V = 0.
C
          do 10 i = 1,n
          aa(i,1) = 1.d+0
10        v(1,1) = v(1,1) + y(i,1)
c
          do 20 k = 2,m
          do 20 i = 1,n
          aa(i,k) = x(i,1) * aa(i,k-1)
20        v(k,1) = v(k,1) + aa(i,k) * y(i,1)
c
          MFIT8 = MSYMVT( MTRAN8(AA,n,m)*AA, m) * V
          return
          end
```

```
APPENDIX B:

*
      Subroutine Main0
*
C ?   y1 m1 d1 y2 m2 d2 cfile(loctb)
*
*     To obtain coefficients for the regression equation for
*     predicting orbital periods
*
      Implicit real*8 (d)
      character*8 cfile,cpp
      Matrix*8 (f)MFIT8(3), MCOFS(3), AA(200,3),VV(3,3)
      dimension dpers(200),dts(200)
*
      common/vbl/dtepoc,elems(6)
      common/tab/tabuf(33)
*
*     xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
      call encode( '(132x, t1,
     1  'Entering period on initiator ',i3/)', tabuf, luc(-23))
      dtgo = dabtim(ipp(1,86), ipp(2,1), ipp(3,1), 0,0,0)
      dtstop = dabtim(ipp(4,1),ipp(5,1),ipp(6,1),0,0,0)
      cfile = cpp(7, 'loctb   ')
      call encode( '('  file: ', a8/)', tabuf, cfile)
      jp = 0
      call getble(cfile, dtgo)
      if(dtgo .lt. dtepoc) dtgo = dtepoc + .01d+0
      dper = 1.701361111d+0/24.d+0
      dtlast = dtepoc
      dt = dtgo
      dpmin = 1.d+20
      dpmax = -dpmin
*
      do 100 n = 1,1000000
      if(dt .gt. dtstop) go to 102
      call getble( cfile, dt)
      if(dtepoc .lt. 0.d+0) go to 102
      if(dtepoc .le. dtlast) go to 100
*
      dlapse = dtepoc - dtlast
      norbs = dlapse/dper + .5d+0
      jp = jp + 1
      dpers(jp) = dlapse/dfloat(norbs)
      if(dpers(jp) .gt. dpmax) dpmax = dpers(jp)
      if(dpers(jp) .lt. dpmin) dpmin = dpers(jp)
      dts(jp) = dtepoc - dtgo
      dtlast = dtepoc
100   dt = dt + 1.d+0
*
102   dymean = .5d+0 * (dpmax + dpmin)
      call initpl(0,0)
```

```
      xscale = 500.d+0 / (dtepoc - dtso)
      yscale = 400.d+0 / (dpmax - dpmin)
*
      do 110 j = 1,jp
      jyp = 250. - yscale*(dpers(j) - dymean)
      jxp = 50. + xscale*dts(j)
      call plot( jyp,jxp,0)
110   call plot(jyp-1, jxp, 3)
*
      do 112 j = 1,jp
112   call encode( '( i6, 1h., 4x, d20.10, d16.8/)', tabuf,
     1  j, dts(j), dpers(j))
*
      MCOFS = MFIT8(dts, dpers, jp, 3,  VV,AA)
*     The matrix MCOFS contains the desired coefficients.
      kf = 0
*
*     Plot the resulting curve over the scatter.
      do 120 j = 1,20
      jxp = 50. + 25.*float(j-1)
      dpx = float(jxp - 50)/xscale
      dpy = MCOFS(1,1) + dpx*(MCOFS(2,1) + dpx*MCOFS(3,1))
      jyp = 250. - yscale*(dpy - dymean)
      call plot(jyp,jxp,kf)
120   kf = 3
*
      call encode( '( *3d16.8)', tabuf, MCOFS)
      call wrtext( 420, 50, 7, tabuf, 48, 3)
      call encode( '(/)', tabuf)
      call endplt
      call mfstak(0)
      return
      end
```

```
PAGE    1                               XXN

APPENDIX C:


             VECTOR FUNCTION VSADJ*4 (dt,dtorig,dcp,correc)
*
*            To compute predicted satellite positions, maing adjustment for
*            gradual changes in orbital period
*
             Implicit real*8 (d), VECTOR*4(V)
             Parameter( halfpi = .5 * 3.141593)
*
             character*8 correc
             MATRIX*8 (f)MFIT8(3), VV(3,3), AA(35,3), MADCOF(3)
             MATRIX*4 (f)MTRAN4(3,3),ROT(3,3),MXR(3,3)
             VECTOR*4 (f)VEC4,(f)VUNIT4,(f)VBLMOD
             dimension dangs(35),dxs(35),dcp(3)
*
             common/vbl/ dtepoc, elems(6)
             common/tab/tabuf(33)
*
             equivalence(vx(1),mxr(1,1)),(vy(1),mxr(1,2)),(vop(1),mxr(1,3))
*
             data dt1,dt2/ 2 * 0.d+0/, delast/0.d+0/, ROT/ 8*0., 1./
c            xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
             if(correc .eq. 'off') go to 30
             dx = dt - dtepoc
             dx2 = dx*dx
             if( delast .eq. dtepoc) go to 30
*
*            Get a new estimate of orbital period.
             dtx = dtepoc - dtorig
             if(dtx .lt. 0.d+0) dtx = 0.d+0
             dper = dcp(1) + dtx*(dcp(2) + dtx*dcp(3))   ;  estimated period
             call tinver(dtepoc, jy,jm,jd,kh,km,ks,nth)
*
c            Compute lead or lag in future Equator crossings
             do 26 j = 1,35
             dtx = dtepoc + 10.d+0 * dper * dfloat(j-1)
             VBLX = VBLMOD(dtx, slat,slong,cd)   ;  Sat pos near Eq crossing
             VOP = VBLX ** VBLMOD(dtx+.01d+0, slat1,slong,cd)
             VEQ = VEC4( vop(1), vop(2), 0.)
             VX = VEQ ** VOP * sign(1., elems(3)-halfpi)
             dxs(j) = dtx - dtepoc   ;  times after epoch
26           dangs(j) = angbtw( VX,VBLX) * sign(1., vblx(3))
*
*            ...and get regression coefficients to estimate them.
             MADCOF = -MFIT8( dxs,dangs,35,3,  VV,AA)
*            call encode( '( 6i3, d16.8, d12.4, 2f9.1/ 1h , *3d16.8/)',
*           1  tabuf, jy,jm,jd,kh,km,ks,dper,dangs(35),slat,vblx(3),
*           2  MADCOF)
*
30           VSAT = VBLMOD(dt, slat,slong,cd)   ;   unadjusted position
```

```
*     .

      if(correc .eq. 'on') then
*
      if(dt.lt.dt1 .or. dt.gt.dt2) then   ;   recompute vector orbit
      dt1 = dt
      dt2 = dt + .02d+0
      VOP = VUNIT4( VBLMOD(dt1, slat1,slong1,cd1) **
     1  VBLMOD(dt2, slat2,slong2,cd2))   ;   unit vector orbital plane
      end if
*
      adj = madcof(1,1) + dx*(madcof(2,1) + dx*madcof(3,1))
*     Angular adjustment which must be made to satellite position
      VEQ = VEC4( vop(1), vop(2), 0.)
      VX = VUNIT4( VEQ ** VOP) * sign(1., elems(3)-halfpi)
      VY = VOP ** VX   ;   the matrix MXR is now defined by equivalence
*     MXR converts a vector from celestial to orbital plane coords
      adj = dx2*adj/(dx2 + 6.d+0)   ;   weights the adjustment
      rot(1,1) = cosine(adj)
      rot(2,1) = sine(adj)
      rot(1,2) = -rot(2,1)
      rot(2,2) = rot(1,1)
      VADJPO = ROT * MTRAN4(MXR,3,3) * VSAT
*     Adjusted position vector in the plane of the orbit
      VSADJ = MXR * VADJPO   ;   rotate back to celestial coords
*
      else
      VSADJ = VSAT
      end if
*
      delast = dtepoc
      return
      end
*
```

```
//VBLM5560 JOB CLASS=B,MSGLEVEL=(0,0)
//* TRVBLMOD N     03/13/86; MOVED IMPLICIT IN BROLYD
//* TRVBLMOD N     03/06/86; NSC
//* (N)     THIS PROGRAM IS IN THE PUBLIC DOMAIN
//*
//A EXEC MFSUB,OPT=2,COND.EDIT=(0,LT),LANGLVL=77
//FORT.SYSPRINT DD SYSOUT=A
//FORT.SYSIN DD *
C
C     VECTOR SATELLITE POSITION BY BROUWER-LYDDANE MODEL
C
C     VECTOR FUNCTION VBLMOD*4 (DTIME,   FLAT,FLONG,CD)
      SUBROUTINE VBLMOD(DTIME, FLAT,FLONG,CD,    VEC)
C
C     'DTIME' IS THE TIME FOR WHICH A SATELLITE POSITION IS DESIRED,
C     EXPRESSED IN JULIAN DAY NUMBER, WHICH IS NOT, REPEAT, IS NOT
C     THE DAY OF THE YEAR.  'DTIME' MAY BE OBTAINED FROM ORDINARY
C     TIME UNITS (YEAR,MONTH,DAY,HOUR,MINUTE,SECOND) USING THE
C     REAL*8 FUNCTION 'DABTIM'.
C
*     FOR A DEFINITION OF 'JULIAN DAY NUMBER' SEE BOWDITCH,
*    'AMERICAN PRACTICAL NAVIGATOR', PUBLICATION NO. 9,  VOL II,
*     P. 469, (DEFENSE MAPPING AGENCY)
*
C     THE RETURNED VALUES ARE GEOCENTRIC LATITUDE, LONGITUDE,
C     CENTRAL DISTANCE (I.E. FROM CENTER OF EARTH TO SATELLITE,
C     IN KILOMETERS), AND THE CELESTIAL POSITION VECTOR.  THE LATTER
C     IS RETURNED THRU THE FUNCTION NAME IF THIS ROUTINE IS CALLED
C     AS A METEFOR VECTOR-VALUED FUNCTION.
C
C     THE COMMON BLOCK /BLXTRA/ RETURNS THE B-L ELEMENTS UPDATED
C     TO THE GIVEN INPUT TIME.
C
C     THE ORBITAL PARAMETERS MUST ALREADY HAVE BEEN PUT INTO THE
C     COMMON BLOCK /VBL/ BY SOME OTHER ROUTINE, SUCH AS 'VBLSET'
C     OR 'GETBLE'.
C
C     THIS ROUTINE USES THE SAME ORBITAL PREDICTION SOFTWARE USED BY
C     NOAA/NESDIS, AND HENCE CANNOT FAIL TO BE UTTERLY CORRECT.
C
C     DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
C
      PARAMETER ( DTR = 6.2831853071795860+0/360.D+0)
C
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*4 GIVENS,VEC(3),VLL(3),FLAT,FLONG,CD
C
      DIMENSION  BLLAST(6), OSCUL(6),AUX(5),
     1  VECPOS(3)
C
      COMMON/BLCNST/ DTSECS,CENDIS,RADEAR,GRAV, DARGS(7)
      COMMON/VBL/ DTEPOC, GIVENS(6)
```

```fortran
      COMMON/BLXTRA/ DTPREP,DBLPRE(6),VECVEL(3)
C
      DATA DTLAST, BLLAST/ -1.D+0, 6*0.D+0/, ANOM/-99999./
C     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
C
      IF(DTIME .LT. DTEPOC) GO TO 900
C
      DO 10 J = 1,6
 10   BLLAST(J) = GIVENS(J)
C
      DTSECS = 86400.D+0 * (DTIME - DTEPOC)
      IF(GIVENS(6).NE.ANOM) CALL BROLYD( OSCUL, BLLAST, 0, 1, 1, AUX)
C     RE-INITIALIZES PREDICTION SOFTWARE IF WE HAVE CHANGED ORBITAL
C     PARAMETERS SINCE THE PRECEDING CALL.
C
      ANOM = GIVENS(6)
      CALL BROLYD( OSCUL, BLLAST, 2, 2, 1, AUX)
C     BROLYD COMPUTES OSCULATING KEPLERIAN VALUES VALID AT TIME
C     DTIME FROM B-L ELEMENTS VALID AT EPOCH 'DTEPOC'.
C
C     RETURN UPDATED B-L ELEMS THRU COMMON /BLXTRA/
      DO 20 J = 1,6
 20   DBLPRE(J) = BLLAST(J)
C
      DTPREP = DTIME
C     EPOCH OF PREDICTED PARAMETERS
C
      CALL CELEM(OSCUL, GRAV, VECPOS, VECVEL)
C     CELEM CONVERTS OSCULATING TO CELESTIAL
C
      DO 30 J = 1,3
 30   VEC(J) = VECPOS(J)
C
      CALL VCOORD( DTIME, VEC, 'CLL ', VLL)
      CD = CENDIS
      FLAT = VLL(1)
      FLONG = VLL(2)
      RETURN
C
 900  FLAT = -999999.
      CD = 0.
      VEC(1) = 0.
      VEC(2) = 0.
      VEC(3) = 0.
      RETURN
      END
C     RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
C
      SUBROUTINE BROLYD
     1 (OSCELE,DPELE,IPERT,IPASS,IDMEAN,ORBEL)
C************************************************************************
C*  REF. '' BROUWER-LYDDANE ORBIT GENERATOR ROUTINE''                  *
```

```
C*                (X-553-70-223)                                       *
C*              BY E.A. GALBREATH 1970                                 *
C*---------------------------------------------------------------------*
C*  MODIFIED 7/31/74 VIONA BROWN AND R.A. GORDON TO INTERFACE WITH GTDS*
C     FURTHER MODIFIED 11 DEC 85 BY F W NAGLE, NOAA/NESDIS, TO
C     PARAMETERIZE THE FRACTIONAL CONSTANTS FOR GREATER SPEED.
C***********************************************************************
      IMPLICIT REAL*8(A-H,O-Z)
C
      PARAMETER(
     X  F3D8=3.0D0/8.0D0,
     X  F1D2=1.0D0/2.0D0,
     X  F3D2=3.0D0/2.0D0,
     X  F1D4=1.0D0/4.0D0,
     X  F5D4=5.0D0/4.0D0,
     X  F1D8=1.0D0/8.0D0,
     X  F5D12=5.0D0/12.0D0,
     X  F1D16=1.0D0/16.0D0,
     X  F15D16=15.0D0/16.0D0,
     X  F5D24=5.0D0/24.0D0,
     X  F3D32=3.0D0/32.0D0,
     X  F15D32=15.0D0/32.0D0,
     X  F5D64=5.0D0/64.0D0,
     X  F35384=35.0D0/384.0D0,
     X  F35576=35.0D0/576.0D0,
     X  F35D52=35.0D0/1152.0D0,
     X  F1D3=1.0D0/3.0D0,
     X  F5D16=5.0D0/16.0D0)
C
C
      DIMENSION OSCELE(6),DPELE(6),ORBEL(5)
      COMMON /BLCNST/ TTO,R,AE,GM,BJ2,BJ3,BJ4,BJ5,FLTINV,XKE,ESQ
C
      DATA BMU, RE/1.0D+0, 1.0D+0/, BKSUBC/0.01D+0/
      DATA PI2/6.283185307179586D+0/
C
C     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
      EK = DSQRT(GM/AE**3)
      DELT = EK*TTO
      GO TO (10,111 ), IPASS
CI-----------------------------I
CI EPOCH ELEMENTS AT EPOCH TIME I
CI-----------------------------I
   10 ADP = DPELE(1)/AE
      EDP = DPELE(2)
      BIDP = DPELE(3)
      HDP = DPELE(4)
      GDP = DPELE(5)
      BLDP = DPELE(6)
      A0 = ADP
      E0 = EDP
      BI0 = BIDP
```

```
      H0 = HDP
      G0 = GDP
      BL0 = BLDP
      IFLG = 0
C--------------------I
C COMPUTE MEAN MOTION I
C--------------------I
      ANU=DSQRT(BMU/A0**3)
C-------------------I
C COMPUTE FRACTIONS I
C-------------------I
      BK2 = -F1D2*(BJ2*RE*RE)
      BK3 = BJ3*RE**3
      BK4 = F3D8*(BJ4*RE**4)
      BK5 = BJ5*RE**5
      GO TO 153
C
111   IF(IPERT.EQ.0)GO TO 7
C
      IF(IDMEAN.NE.0) GO TO 202
C
      ADP = DPELE(1)/AE
      EDP = DPELE(2)
      BIDP = DPELE(3)
      HDP = DPELE(4)
      GDP = DPELE(5)
      BLDP = DPELE(6)
C
  153 EDP2=EDP*EDP
      CN2=1.0-EDP2
      CN23 = CN2*CN2*CN2
      CN=DSQRT(CN2)
      GM2=BK2/ADP**2
      GMP2=GM2/(CN2*CN2)
      GM4=BK4/ADP**4
      GMP4=GM4/CN**8
      THETA=DCOS(BIDP)
      THETA2=THETA*THETA
      THETA4=THETA2*THETA2
C
  202 IF(IDMEAN.EQ.0)GO TO 155
      IF ( IPASS.EQ.2 ) GO TO 150
C------------------------I
C COMPUTE LDOT,GDOT,HDOT I
C------------------------I
  157 BLDOT=CN*ANU*(GMP2*(F3D2*(3.0*THETA2-1)+GMP2*F3D32*(THETA2
     1*(-96.0*CN+30.0-90.0*CN2)+(16.0*CN+25.0*CN2-15.0)+THETA4
     2*(144.0*CN+25.0*CN2+105.0)))+EDP2*GMP4*F15D16*(3.0+35.0*THETA4
     3-30.0*THETA2))
      GDOT=ANU*(F5D16*GMP4*((THETA2*(126.0*CN2-270.0)+THETA4*(385.0
     1-189.0*CN2))-9.0*CN2+21.0)+GMP2*(F3D32*GMP2*(THETA4*(45.0*CN2
     2+360.0*CN+385.0)+THETA2*(90.0-192.0*CN-126.0*CN2)+(24.0*CN
```

```
     3+25.0*CN2-35))+F3D2*(5*THETA2-1)))
      HDCT=ANU*(GMP4*F5D4*THETA*(3.0-7.0*THETA2)*(5.0-3.0*CN2)+GMP2
     1*(GMP2*F3D8*(THETA*(12.0*CN+9.0*CN2-5.0)-THETA*THETA2*(5.0*CN2
     2+36.0*CN+35.0))-3*THETA))
  155 IF(IFLG.EQ.1)GO TO 19
CI--------------------------------------------I
CI COMPUTE ISUBC TO TEST CRITICAL INCLINATION I
CI--------------------------------------------I
      BISUBC=((1.0-5.0*THETA2)**(-2))*((25.0*THETA4*THETA)*(GMP2*EDP2))
      IFLG=1
CI--------------------------------------I
CI FIRST CHECK FOR CRITICAL INCLINATION I
CI--------------------------------------I
      IF(BISUBC.GT.BKSUBC)GO TO 158
      ASSIGN 163 TO ID8
      GO TO 159
C-------------------------------I
C IS THERE CRITICAL INCLINATION I
C-------------------------------I
   19 IF(BISUBC.GT.BKSUBC)GO TO 150
  159 IF(IPERT.EQ.1)GO TO 150
      GM3=BK3/ADP**3
C     GMP3=GM3/(CN2*CN2*CN2)
      GMP3=GM3/CN23
      GM5=BK5/ADP**5
      GMP5=GM5/CN**10
      G3DG2=GMP3/GMP2
      G4DG2=GMP4/GMP2
      G5DG2=GMP5/GMP2
CI---------------I
CI COMPUTE A1-A8 I
CI---------------I
      A1=(F1D8*GMP2*CN2)*(1.0-11.0*THETA2-((40.0*THETA4)/(1.0-5.0*THETA2
     1)))
      A2=(F5D12*G4DG2*CN2)*(1.0-((8.0*THETA4)/(1.0-5.0*THETA2))-3.0
     1*THETA2)
      A3=G5DG2*((3.0*EDP2)+4.0)
      A4=G5DG2*(1.0-(24.0*THETA4)/(1.0-5.0*THETA2)-9.0*THETA2)
      A5=(G5DG2*(3.0*EDP2+4.0))*(1.0-(24.0*THETA4)/(1.0-5.0*THETA2)-9.0
     1*THETA2)
      A6=G3DG2*F1D4
      SINI=DSIN(BIDP)
      A10=CN2*SINI
      A7=A6*A10
      A8P=G5DG2*EDP*(1.0-(16.0*THETA4)/(1.0-5.0*THETA2)-5.0*THETA2)
      A8=A8P*EDP
C
C     COMPUTE B13-B15
C
      B13=EDP*(A1-A2)
      B14=A7+F5D64*A5*A10
      B15=A8*A10*F35384
```

```
C
C    COMPUTE A11-A27
C
      A11=2.0+EDP2
      A12=3.0*EDP2+2.0
      A13=THETA2*A12
      A14=(5.0*EDP2+2.0)*(THETA4/(1.0-5.0*THETA2))
      A17=THETA4/((1.0-5.0*THETA2)*(1.0-5.0*THETA2))
      A15=(EDP2*THETA4*THETA2)/((1.0-5.0*THETA2)*(1.0-5.0*THETA2))
      A16=THETA2/(1.0-5.0*THETA2)
      A18=EDP*SINI
      A19=A18/(1.0+CN)
      A21=EDP*THETA
      A22=EDP2*THETA
      SINI2=DSIN(BIDP/2.0)
      COSI2=DCOS(BIDP/2.0)
      TANI2=DTAN(BIDP/2.0)
      A26=16.0*A16+40.0*A17+3.0
      A27=A22*F1D8*(11.0+200.0*A17+80.0*A16)
CI-----------------I
CI COMPUTE B1-B12 I
CI-----------------I
      B1=CN*(A1-A2)-((A11-400.0*A15-40.0*A14-11.0*A13)*F1D16+(11.0+200.0
     1*A17+80.0*A16)*A22*F1D8)*GMP2+((-80.0*A15-8.0*A14-3.0*A13+A11)
     2*F5D24+F5D12*A26*A22)*G4DG2
      B2=A6*A19*(2.0+CN-EDP2)+F5D64*A5*A19*CN2-F15D32*A4*A18*CN*CN2
     1+(F5D64*A5+A6)*A21*TANI2+(9.0*EDP2+26.0)*F5D64*A4*A18+F15D32*A3
     2*A21*A26*SINI*(1.0-THETA)
      B3=((80.0*A17+5.0+32.0*A16)*A22*SINI*(THETA-1.0)*F35576*G5DG2*EDP)
     1-((A22*TANI2+(2.0*EDP2+3.0*(1.0-CN2*CN))*SINI)*F35D52*A8P)
      B4=CN*EDP*(A1-A2)
      B5=((9.0*EDP2+4.0)*A10*A4*F5D64+A7)*CN
      B6=F35384*A8*CN2*CN*SINI
      B7=((CN2*A18)/(1.0-5.0*THETA2))*(F1D8*GMP2*(1.0-15.0*THETA2)+(1.0
     1-7.0*THETA2)*G4DG2*(-F5D12))
      B8=F5D64*(A3*CN2*(1.0-9.0*THETA2-(24.0*THETA4/(1.0-5.0*THETA2))))
     1+A6*CN2
      B9=A8*F35384*CN2
      B10=SINI*(A22*A26*G4DG2*F5D12-A27*GMP2)
      B11=A21*(A5*F5D64+A6+A3*A26*F15D32*SINI*SINI)
      B12=-((80.0*A17+32.0*A16+5.0)*(A22*EDP*SINI*SINI*F35576*G5DG2)+(A8
     1*A21*F35D52))
  150 IF(IPERT.EQ.0)GO TO 7
      IF(IDMEAN.EQ.0)GO TO 4
C-----------------------I
C COMPUTE SECULAR TERMS I
C-----------------------I
CI-----------------------I
CI ''MEAN'' MEAN ANOMALY I
CI-----------------------I
      BLDP = ANU*DELT + BLDOT*DELT + BLO
      BLDP = DMOD(BLDP,PI2)
```

```
      IF(BLDP.LT.0.0D0)BLDP = BLDP + PI2
CI----------------------------I
CI   MEAN ARGUMENT OF PERIGEE I
CI----------------------------I
      GDP = GDOT*DELT  + G0
      GDP = DMOD(GDP,PI2)
      IF(GDP.LT.0.0D0)GDP = GDP + PI2
C    MEAN LONGITUDE OF ASCENDING NODE
      HDP = HDOT*DELT + H0
      HDP = DMOD(HDP,PI2)
      IF(HDP.LT.0.0D0)HDP = HDP + PI2
C
    4 DO 33 NN=1,6
   33 OSCELE(NN) = DPELE(NN)
C
      A = ADP
      E = EDP
      BI = BIDP
      H = HDP
      G = GDP
      BL = BLDP
CI-------------------------------------I
CI COMPUTE TRUE ANOMALY(DOUBLE PRIMED) I
CI-------------------------------------I
      EADP = DKEPLR(BLDP,EDP)
      SINDE= DSIN(EADP)
      COSDE= DCOS(EADP)
      SINFD= CN*SINDE
      COSFD= COSDE - EDP
      FDP = DATAN0(SINFD,COSFD)
      IF(IPERT.EQ.1) GO TO 7
C
C     DADR=(1.0-EDP*COSDE)**(-1)
      DADR = 1.D+0 / (1.D+0 - EDP*COSDE)
      SINFD=SINFD*DADR
      COSFD=COSFD*DADR
      CS2GFD=DCOS(2.0*GDP+2.0*FDP)
      DADR2=DADR*DADR
      DADR3=DADR2*DADR
      COSFD2=COSFD*COSFD
CI----------------------------I
CI COMPUTE A(SEMI-MAJOR AXIS) I
CI----------------------------I
      A=ADP*(1.0+GM2*((3.0*THETA2-1.0)*(EDP2/CN23)*(CN+(1.0/(1.
     1+CN)))+((3.0*THETA2-1.0)/CN23)*(EDP*COSFD)*(3.0+3.0*EDP
     2*COSFD+EDP2*COSFD2)+3.0*(1.0-THETA2)*DADR3*CS2GFD))
      SN2GFD=DSIN(2.0*GDP+2.0*FDP)
      SNF2GD=DSIN(2.0*GDP+FDP)
      CSF2GD=DCOS(2.0*GDP+FDP)
      SN2GD=DSIN(2.0*GDP)
      CS2GD=DCOS(2.0*GDP)
      SN3GD=DSIN(3.0*GDP)
```

```
      CS3GD=DCOS(3.0*GDP)
      SN3FGD=DSIN(3.0*FDP+2.0*GDP)
      CS3FGD=DCOS(3.0*FDP+2.0*GDP)
      SINGD=DSIN(GDP)
      COSGD=DCOS(GDP)
      GO TO ID8,(163,164)
  163 DLT1E=B14*SINGD+B13*CS2GD-B15*SN3GD
CI------------------------I
CI COMPUTE (L+G+H) PRIMED I
CI------------------------I
      BLGHP=HDP+GDP+BLDP+B3*CS3GD+B1*SN2GD+B2*COSGD
      BLGHP=DMOD(BLGHP,PI2)
      IF(BLGHP.LT.0.0D0)BLGHP=BLGHP+PI2
      EDPDL=B4*SN2GD-B5*COSGD+B6*CS3GD-F1D4*CN2*CN*GMP2*(2.0*(3.0*THETA2
     1-1.0)*(DADR2*CN2+DADR+1.0)*SINFD+3.0*(1.0-THETA2)*((-DADR2*CN2
     2-DADR+1.0)*SNF2GD+(DADR2*CN2+DADR+F1D3)*SN3FGD))
       DLTI=F1D2*THETA*GMP2*SINI*(EDP*CS3FGD+3.0*(EDP*CSF2GD+CS2GFD))
     1-(A21/CN2)*(B8*SINGD+B7*CS2GD-B9*SN3GD)
       SINDH=(1.0/COSI2)*(F1D2*(B12*CS3GD+B11*COSGD+B10*SN2GD-(F1D2*GMP2
     1*THETA*SINI*(6.0*(EDP*SINFD-BLDP+FDP)-(3.0*(SN2GFD+EDP*SNF2GD)+EDP
     2*SN3FGD)))))
CI-----------------I
CI COMPUTE (L+G+H) I
CI-----------------I
  164 BLGH=BLGHP+((1.0/(CN+1.0))*F1D4*EDP*GMP2*CN2*(3.0*(1.0-THETA2)*
     1(SN3FGD*(F1D3+DADR2*CN2+DADR)+SNF2GD*(1.0-(DADR2*CN2+DADR)))+2.0*
     2SINFD*(3.0*THFTA2-1.0)*(DADR2*CN2+DADR+1.0)))+GMP2*F3D2*((-2.0*
     3THETA-1.0+5.0*THETA2)*(EDP*SINFD+FDP-BLDP))+(3.0+2.0*THETA-5.0*
     4THETA2)*(GMP2*F1D4*(EDP*SN3FGD+3.0*(SN2GFD+EDP*SNF2GD)))
       BLGH=DMOD(BLGH,FI2)
       IF(BLCH.LT.0.0D0)BLGH=BLGH+PI2
       DLTE=DLT1E+(F1D2*CN2*((3.0*(1.0/CN23)*GM2*(1.0-THETA2)
     1*CS2GFD*(3.0*EDP*COSFD2+3.0*COSFD+EDP2*COSFD*COSFD2+EDP))-(GMP2
     2*(1.0-THETA2)*(3.0*CSF2GD+CS3FGD))+(3.0*THETA2-1.0)*GM2*(1.0/
     3CN23)*(EDP*CN+(EDP/(1.0+CN))+3.0*EDP*COSFD2+3.0*COSFD+
     4EDP2*COSFD*COSFD2)))
       EDPDL2=EDPDL*EDPDL
       EDPDE2=(EDP+DLTE)*(EDP+DLTE)
CI--------------------------I
CI COMPUTE F(ECCENTRICITY) I
CI--------------------------I
      E=DSQRT(EDPDL2+EDPDE2)
      SINDH2=SINDH*SINDH
      SQUAR=(DLTI*COSI2*F1D2+SINI2)*(DLTI*COSI2*F1D2+SINI2)
      SQRI=DSQRT(SINDH2+SQUAR)
CI----------------------------I
CI COMPUTE BI (INCLINATION) I
CI----------------------------I
      BI=DARSIN(SQRI)
      BI=2.0*BI
      BI=DMOD(BI,PI2)
      IF(BI.LT.0.0D0)BI=BI+PI2
```

```
CI-----------------------------I
CI CHECK FOR E(ECCENTRICITY)=0 I
CI-----------------------------I
      IF(E.NE.0.0) GO TO 168
      BL=0.0
CI-----------------------------I
CI CHECK FOR BI(INCLINATION)=0 I
CI-----------------------------I
  145 IF(BI.NE.0.0) GO TO 169
      H=0.0
CI---------------------------------I
CI COMPUTE G(ARGUMENT  OF PERIGEE) I
CI---------------------------------I
  146 G=BLGH-BL-H
      G=DMOD(G,PI2)
      IF(G.LT.0.0D0)G=G+PI2
CI----------------------I
CI COMPUTE TRUE ANOMALY I
CI----------------------I
      EA    = DKEPLR(BL,E)
      ARG1 = DSIN(EA) * DSQRT(1.0-E**2)
      ARG2 = DCOS(EA) - E
      F = DATANO(ARG1,ARG2)
C
      OSCELE(1) = A*AE
      OSCELE(2) = E
      OSCELE(3) = BI
      OSCELE(4) = H
      OSCELE(5) = G
      OSCELE(6) = BL
C
7     DPELE(1) = ADP*AE
      DPELE(2) = EDP
      DPELE(3) = BIDP
      DPELE(4) = HDP
      DPELE(5) = GDP
      DPELE(6) = BLDP
      IF(IPERT.EQ.0)BL = DMOD(ANU*DELT,PI2)
      ORBEL(1) = EADP
      ORBEL(2) = GDP + FDP
      ORBEL(3) = GDP
      ORBEL(4) = EK*(ANU + BLDOT)
      ORBEL(5) = FDP
      R = A*AE*(1.0D0 - E*DCOS(EA))
      GO TO 45
CI----------------------------------------I
CI MODIFICATIONS FOR CRITICAL INCLINATION I
CI----------------------------------------I
  158 DLT1E=0.0
      BLGHP=0.0
      EDPDL=0.0
      DLTI=0.0
```

```
      SINDH=0.0
      ASSIGN 164 TO ID8
      GO TO 150
  168 SINLDP=DSIN(BLDP)
      COSLDP=DCOS(BLDP)
      SINHDP=DSIN(HDP)
      COSHDP=DCOS(HDP)
CI-------------------------I
CI COMPUTE L(MEAN ANOMALY) I
CI-------------------------I
      ARG1=EDPDL*COSLDP+(EDP+DLTE)*SINLDP
      ARG2=(EDP+DLTE)*COSLDP-(EDPDL*SINLDP)
      BL=DATAN2(ARG1,ARG2)
      BL=DMOD(BL,PI2)
      IF(BL.LT.0.0D0)BL=BL+PI2
      GO TO 145
CI-----------------------------------------I
CI COMPUTE H(LONGITUDE OF ASCENDING NODE) I
CI-----------------------------------------I
  169 ARG1=SINDH*COSHDP+SINHDP*(F1D2*DLTI*COSI2+SINI2)
      ARG2=COSHDP*(F1D2*DLTI*COSI2+SINI2)-(SINDH*SINHDP)
     H=DATAN2(ARG1,ARG2)
     H=DMOD(H,PI2)
     IF(H.LT.0.0D0)H=H+PI2
      GO TO 146
   45 CONTINUE
      RETURN
      END
C
C     RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
      SUBROUTINE CELEM (ORBEL,GMC,PV,VV)
C   ORIGINAL VERSION ...1/22/71....CHARLES K. CAPPS
C   PURPOSE:
C       THIS ROUTINE CONVERTS CLASSICAL OSCULATING ORBITAL ELEMENTS
C       TO CARTESIAN ELEMENTS.
C   CALLING SEQUENCE:
C       CALL CELEM(ORBEL,GMC,PV,VV)
C   INPUT THRU ARGUMENT LIST:
C       ORBEL(1) = SEMI-MAJOR AXIS,A  (OSCULATING ELEMENTS)
C       ORBEL(2) = ECCENTRICITY, E
C       ORBEL(3) = INCLINATION, I
C       ORBEL(4) = LONGITUDE OF ASCENDING NODE, CAP OMEGA
C       ORBEL(5) = ARGUMENT OF PERIFOCUS, OMEGA
C       ORBEL(6) = MEAN ANOMALY, M
C       GMC = GRAVITATIONAL CONSTANT
C   OUTPUT THRU ARGUMENT LIST:
C       PV = CARTESIAN POSITION VECTOR
C       VV = CARTESIAN VELOCITY VECTOR
C   METHOD:
C       USES MILES STANDISH ITERATIVE SCHEME FOR SOLN TO KEPLERS EQN.
C   REFERENCES:
C       GTDS TASK SPEC FOR CELEM, C.E. VELEZ, 13 JANUARY 1971
```

```
C       DODS SYSTEM DESCRIPTION. SUBROUTINE KEPLR1
C       R. EXCOBAL-'METHODS OF ORBIT DETERMINATION'
C       X-552-67-421,'COMPARISON FO ITERATIVE TECHNIQUES FOR THE
C       SOLUTION OF KEPLERS EQUATION', I.COLE AND R. BORCHERS
C   PROGRAMMER:
C       CHARLES K. CAPPS,  CODE 553.2,  GSFC
C
      IMPLICIT REAL*8(A-H,O-Z)
C
      DATA MAX /10/
      DIMENSION PV(3),VV(3),ORBEL(6)
      DATA TOL /+0.5D-16/
C
      ITER = 0
C     FIND IF THIS IS ELLIPTIC OR HYPERBOLIC ORBIT
      IF (ORBEL(1).LE.0.0D0.AND.ORBEL(2).GT.1.0D0) GO TO 50
C
C     ELLIPTIC ORBIT TAKES THIS ROUTE.
C     FIRST FIND ECCENTRIC ANOMALY VIA NEWTONS (MILES STANDISH VERSION)
      E1 = ORBEL(6)
  10  F = E1 - (ORBEL(2) * DSIN(E1)) - ORBEL(6)
      D = 1.0D0 - (ORBEL(2) * DCOS(E1 - 0.5D0 * F))
      E2 = E1 - (F / D)
      IF (DABS (E1-E2)-TOL )40,40,20
   20 ITER = ITER + 1
      E1 = E2
      IF(ITER - MAX) 10,10,30
C     SET UP ERROR CODE TO RETURN FROM SUBROUTINE
   30 NERR = 13
C     ECCENTRIC ANOMALY CONVERGED, NOW GET XO, YO, R
   40 COSE = DCOS(E2)
      SINE = DSIN (E2)
      TEMP = 1.0D0 - ORBEL(2) * ORBEL(2)
      XO = ORBEL(1) * (COSE - ORBEL(2))
      YO = ORBEL(1) * (DSQRT(TEMP)* SINE)
      R = ORBEL(1) * (1.0D0 - ORBEL(2) * COSE)
      XOD = (-DSQRT(GMC* ORBEL(1))* SINE)/R
      YOD = (DSQRT(GMC*ORBEL(1)*(TEMP))*COSE) / R
      GO TO 100
C
C     HYPERBOLIC ORBITS TAKE THIS ROUTE
   50 E1 = ORBEL(6) / 2.0D0
   60 F = ORBEL(2) * DSINH(E1) - E1 - ORBEL(6)
      D = ORBEL(2) * DCOSH(E1 - 0.5D0 * F ) - 1.0D0
      E2 = E1 - (F / D)
      IF (DABS (E1-E2)-TOL )90,90,70
   70 ITER = ITER + 1
      E1 = E2
      IF (ITER - MAX) 60,60,80
C      SET UP ERROR CODE FOR NON-CONVERGENCE PRIOR TO EXIT.
   80 NERR = 14
C     ECCENTRIC ANOMALY COMPUTED, NOW GET XO,YO,R
```

```
   90 COSE = DCOSH (E2)
      SINE = DSINH(E2)
      TEMP = ORBEL(2) * ORBEL(2) - 1.0D0
      XO = ORBEL(1)*(COSE- ORBEL(2))
      YO = - ORBEL (1)*DSQRT (TEMP) * SINE
      R = ORBEL(1)*(1.0D0 - ORBEL(2) * COSE)
      XOD = (-DSQRT(-GMC*ORBEL(1))*SINE)/R
      YOD = (DSQRT(-GMC*ORBEL(1)*TEMP)*COSE) / R
  100 COSO = DCOS(ORBEL(5))
      SINO = DSIN (ORBEL(5))
      COSOM = DCOS (ORBEL(4))
      SINOM = DSIN (ORBEL(4))
      COSI = DCOS(ORBEL(3))
      SINI = DSIN (ORBEL(3))
      B11 = COSO * COSOM - SINO * SINOM * COSI
      B21 = COSO * SINOM + SINO * COSOM * COSI
      B31 = SINO * SINI
      B12 = -SINO * COSOM - COSO * SINOM * COSI
      B22 = -SINO * SINOM + COSO * COSOM * COSI
      B32 = COSO * SINI
C     NOW MULTIPLY 3 X 2 MARTIX BY 2 X 1 VECTORS FOR POSITION, VELOCITY.
      PV(1) = B11 * XO + B12 * YO
      PV(2) = B21 * XO + B22 * YO
      PV(3) = B31 * XO + B32 * YO
      VV (1) = B11*XOD + B12 * YOD
      VV(2) = B21 * XOD + B22 * YOD
      VV(3) = B31 * XOD + B32 * YOD
  999 RETURN
      END
C
C     RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
      BLOCK DATA
      IMPLICIT REAL*8 (A-H,O-Z)
C
C   NAME- BLCNST
C
C   LANGUAGE- FORTHXP     TYPE- PROGRAM
C
C   THIS COMMON BLOCK WAS UPDATED MARCH 28, 1984 TO INCLUDE XKE AND ESQ
C    BY E. HARROD S/SP12
C   THIS BLOCK DATA IS COMPILED WITH THE ROUTINE PSCEAR, ANY PROGRAM
C    USING PSCEAR DOES NOT NEED TO RECOMPILE THIS BLOCK DATA
C
C*********************************************************************
C
      COMMON/BLCNST/ TTO,R,AE,GM,BJ2,BJ3,BJ4,BJ5,FLTINV,XKE,ESQ
C
      DATA TTO,R,GM,AE,BJ2,BJ3,BJ4,BJ5,FLTINV,XKE,ESQ/2*0.D0,
     * 398600.8D0,6378.135D0,-0.10826158D-02,0.25388100D-05,
     * 0.165597000D-05,0.218482660-06,298.25D0,0.743669161D-01,
     * 0.6994317778266721D-02/
      END
```

```
C
C     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
      REAL FUNCTION DATANO*8 (DA,DB)
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DATA PI2/6.283185307179586D+0/
C
      DA = DATAN2(DA,DB)
      IF(DA .LT. 0.D+0) DA = DA + PI2
      DATANO = DA
      RETURN
      END
C     RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
C
      FUNCTION DKEPLR(M,E)
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 M,PI2/6.283185307179586D0/,TOL/0.5D-15/
C
C         SUBROUTINE TO SOLVE KEPLER'S EQ.
C  KEPLER'S EQ.,RELATES GEOMETRY OR POSITION IN ORBIT PLANE TO TIME.
C
C M - MEAN ANOMALY (0<M<2PI)
C E - ECCENTRICITY
C EA- ECCENTRIC ANOMALY
C
      EA=0
      IF(M)1,2,1
    1 EA=M + E*DSIN(M)
      DO 22 I=1,12
      OLDEA=EA
      FE=EA-E*DSIN(EA)-M
      EA=EA-FE/(1-E*DCOS(EA-0.5D0*FE))
C TEST FOR CONVERGENCE
      DELEA=DABS(EA-OLDEA)
      IF(DELEA.LE.TOL)GO TO 2
   22 CONTINUE
    2 EA=DMOD(EA,PI2)
      DKEPLR=EA
      RETURN
      END
/*
//EDIT.SYSPRINT DD SYSOUT=A
//EDIT.SYSIN DD *
 NAME VBLMOD(R)
/*
//
```

The Technical Proceedings of
The Third International TOVS Study Conference

Madison, Wisconsin

August 13 - 19, 1986

Edited by

W. P. Menzel

Cooperative Institute for Meteorological Satellite Studies
Space Science and Engineering Center
University of Wisconsin
1225 West Dayton Street
Madison, Wisconsin 53706
(608)262-0544

November 1986