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 Innsbruck, 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 mean 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

\[ ep, ep + 10p, ep + 20p, ep + 30p, \text{ etc.} \]

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

\[ VOP = VUNIT4 \; (VSAT1 \; \ast \; \ast \; \ast \; VSAT2) \]

where the double asterisk (\(\ast\)) 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
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

190
the Equator crossings, and the matrix \( \text{ROT} \) which accomplishes this rotation in the orbital plane is

\[
\text{ROT} = \begin{bmatrix}
\text{cosine (adj)} & -\text{sine (adj)} & 0 \\
\text{sine (adj)} & \text{cosine (adj)} & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

The routine \text{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 errors 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.
* 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)

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)MALLB(100), DINELS(6)

* common/blxtra/dtprep,UPDELS,VECVXL; Updated BL elements
common/vbl/ dtepoc, BLELEM

* equivalence( VSAT(1), MVSAT)

read(9,**) jskip ; skip-ahead time in hours
read(9,**) Jy,Jm,Jd,kh,km,sec ; reads definitive epoch
dtepoc = dbtim(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,f8.3/1 h, 3e16.6/
1 1 h, 3e16.6/)

do 2 j = 3,6 ; conv angles to radians
blelem(j,1) = .01745329*blelem(j,1)

dtso = dtepoc + dfloat(jskip)/24.d+0

do 100 n = 1,200 ; find approximate start of ascending pass
  dt = dtso + 3.d+0 * dfloat(n)/1440.d+0
  VWB = VBLMOD(dt, flat,flons,cd)

* if(n .eq. 1) then ; altitude and kinetic energy initially
dkine = 1.d+6 * VECVEL*VECVEL
dzorid = cd

193
dsfa = 9.8d0 * (6371.d0 / dzoris)**2
write(6,91) cd,dkinet
*  
format('Initial altitude and energy ', f9.1,2x,d18.7/
else
  dk = 1.d6 * VECVEL*VECVEL
dp = 1000.d0 *dsfa *(dbl(c) - dzoris)
dtort = dp + dk
write(6,93) n,dk,dp,dtot
*  
format(4x, i6, 3d18.8)
end if
*  
if(flat,lt., -75.) go to 102 ; close to minimum latitude
continue
*  
do 110 n = 1,100; search northward in small steps
  dtt(n+1) = dt + dtsstep * dfloat(n)
  VSAT(n) = VBLOD(dt(n+1), flat,flons,cd)
  c
  VBLMOD also returns updated BL elements thru common/blxtra/
  DELEMS(n,*) = UPDELS
*  
110 DELEMS(n,*) = UPDELS

litsch(MVSAT(3,3), 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.
*  
Jsg = litsch(MVSAT(3,3), 100); choose 5 points on either side near Equator
Jstop = litsch(Jsg, 100); choose 5 points on either side near Equator
mx = mcoefs(Jsg,Jstop,1)
MCOEFS = MFIT8(MX, DTT(Jsg,1), 11, mr, VV, AA)
*  
We just obtained mr coefficients fitting time as a function
of z-component near Equator crossings. The constant term
(the approximation when z=0) is the crossing time.
*  
dt = mcoefs(1,1); interpolated crossing time
   call tinver( dt, 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
format('Approximate crossing time ', 6i3/)
*  
Extract the sub-matrix of orbital parameters straddling Equator
MSUB = MSUB18( DELEMS, 100, 6, Jsg, Jstop+1, 1, 6+1); 11x6
DT = DTT - MALL8(dtx, 100,1); subtract dtx from all times
*  
DTT times are now relative to Equator crossings.
*  
do 150 j = 1, 6; interpolate each of 6 elements to dtx
   MCOEFS = MFIT8(DTT(Jsg+1), MSUB(*,j), 11, mr, VV, AA)
dinels(j+1) = mcoefs(1,1)
*  
convert parameters to McIdas storage format
ecc = dinels(2,1)
ap = drtdx*dinels(5,1)
ra = drtddinel(4,1)
finc1 = drtddinel(3,1)
sma = dinel(1,1)
fma = drtddinel(6,1)
call tinver(dt, jy, jm, jd, kh, km, ks, nth)
frac = 86400.0 * (dt - dabtim(jy, jm, jd, kh, km, ks))
sec = float(kss + frac
write (6,77) jy, jm, jd, kh, km, sec, sma, ecc, finc1, ra, ap, fma
format (1h15.3, f8.2, f8.2, f9.6, f7.3, 3f8.3/
) Compute energies using the definitive elements, and also using
the pseudo-BUS elements just computed.
fx = vblmod(dt, flat, flons, cd)
k = 1, dt6 * veve, veve
f = 1000.0 * dsfa * (dbl cd cd) - dzoris)
dot = k * dt6 + dpt
write (6,83) flat, flons, kin, dop, dot
format (1h, 2f9.1/h, 3d18.8)

dtepc = dt
BLELEM = DINEL’S; insert pseudo-elements just computed.
USU = vblmod(dt, flat, flons, cd)
dkin = 1, dt6 * veve, veve
f = 1000.0 * dsfa * (dbl cd cd) - dzoris)
dot = k * dt6 + dpt
write (6,83) flat, flons, kin, dop, dot
format (1h, 2f9.1/h, 3d18.7)
call mfstak(0)
return
end

******************************************************************************
Matrix Function MFIT8*8 (xy, ym, v, aa)

UPPER case symbols are matrices, lower case are scalars.
To obtain the m coefficients (m,1) for a least-squares fit to
n xy pairs of data. See P. 278 on ‘Approximation’ in Frobers.
The functional value returned by this routine is the (m,1)
matrix of real*8 fitting coefficients, in ascending powers.
If this routine is called from a conventional Fortran program,
the call is:

This routine is similar to MATFIT, but has REAL*8 inputs.

CALL MFIT8(XLIST, YLIST, NPAIRS, ORDER, v, aa, COEFS)

Matrix*8 MFIT8(m, aa(n,m), (f)MTRANS(m,n), (f)MSYMVT(m,m),

195
157  \textbf{1} \quad V(m), X(n), Y(n)
158  \textbf{c}
159  \textbf{C}
160  \textbf{C}
161  \textbf{do} 10 \textbf{i} = 1, n
162  \textbf{a}a(i,1) = 1, d+0
163  \textbf{10} \quad v(1,1) = v(1,1) + y(i,1)
164  \textbf{c}
165  \textbf{do} 20 \textbf{k} = 2, m
166  \textbf{do} 20 \textbf{i} = 1, n
167  \textbf{a}a(i,k) = y(i,1) * a(a(i,k-1)
168  \textbf{20} \quad v(k,1) = v(k,1) + a(a(i,k) * y(i,1)
169  \textbf{c}
170  \textbf{M}FITB = MSYMVT(MTRANB(AA,n,m)\times AA, m) \times V
171  \textbf{return}
172  \textbf{end}
APPENDIX B:

Subroutine Main0

C

To obtain coefficients for the regression equation for predicting orbital periods

Implicit real*8 (d)
character*8 cfile, ccpp
Matrix*8 (f)MFITB(3), MCOFS(3), AA(200,3), VV(3,3)
dimension dpers(200), dts(200)

common/vbl/dtepoc, eles(6)
common/tab/tabuf(33)

*xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
*call encode( ' (132x, t1,
* 1  "Entering period on initiator ', i3/)', tabuf, lcc(-23))
dts0 = 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 = ccpp(7, 'loctb ')
call encode( '(* file: ', a8/)', tabuf, cfile)
jp = 0
call setlc(cfile, dtso)
if(dtso .lt. dtepoc) dtso = dtepoc + .01d0
dper = 1.701361111d0/24.d0
dtlast = dtepoc
dt = dtso
dpmin = 1.d20
dpmax = -dpmin

**
do 100 n = 1,1000000
if(dt .gt. dtstop) go to 102
call setlc(cfile, dt)
if(dtepoc .lt. 0.d0) go to 102
if(dtepoc .le. dtlast) go to 100
**
dlapse = dtepoc - dtlast
norbs = dlapse/dper + .5d0
jp = jp + 1
dpers(jp) = dlapse/dfloat(norbs)
if(dpers(jp) .lt. dpmax) dpmax = dpers(jp)
if(dpers(jp) .lt. dpmin) dpmin = dpers(jp)
dts(jp) = dtepoc - dtso
dtlast = dtepoc
100 dt = dt + 1.d0
**
dymean = .5d0 * (dpmax + dpmin)
call initrp(0,0)
xscale = 500.d+0 / (dtepoc - dtso)
yscale = 400.d+0 / (dpmx - dpmn)

* do 110 j = 1,jp
   jyp = 250. - yscale*(dpers(j) - dymean)
   jxp = 50. + xscale*dts(j)
   call plot(jyp,jxp)
   call plot(jyp-1, jxp, 3)

* do 112 j = 1,jp
   call encode( '(' i6, 1h, , 4x, d20.10, d16.8/)', tabuf,
   1   j, dts(j), dpers(j))
   mcofs = mfit8(dts, dpers, j, 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)
   dp = 1.0d0 * (jxp - 50.0d0) / xscale
   dpy = mcofs(1,1) + dp*x*(mcofs(2,1) + dp*x*mcofs(3,1))
   jyp = 250. - yscale*(dpy - dymean)
   call plot(jyp,jxp,kf)
   kf = 3

* call encode( '(' *3d16.8)', tabuf, MCOFS)
   call wrtext( 420, 50, 7, tabuf, 48, 3)
   call encode( '(' /', tabuf)
   call endplt
   call mfstack(0)
   return
   end
APPENDIX C:

VECTOR FUNCTION VSADJ*4 (dt, dtoris, dcp, correc)

* To compute predicted satellite positions, mains 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) MFI8(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 dans(35), dxs(35), dcp(3)

common /vbl/ dtepoc, elems(6)
common /tabuf/ (33)

* equivalence(vx(1),mxr(1,1)),(vy(1),mxr(1,2)),(vp(1),mxr(1,3))

data dt1, dt2/ 2 * 0.d+0/, delast/0.d+0/, ROT/8.d+0., 1./

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 - dtoris
if(dtx .lt. 0.d+0) dtx = 0.d+0
dper = dcp(1) + dtx*(dcp(2) + dtx*dcp(3)) ! estimated period
call tinvr(dtepoc, Jy, jm, Jd, kh, km, ks, nth)

* 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 pr near Eq crossings
VOP = VBLX ** VBLMOD(dtx+0.d+0, slat, slong, cd)
VEQ = VEC4( vop(1), vop(2), 0., 0.)
VX = VEQ ** VOP * sign(1., elems(3)-halfpi)
dxs(J) = dtx - dtepoc ! times after epoch

dans(J) = ansbtw (VX, VBLX) * sign(1., vbix(3))

* ...and set regression coefficients to estimate them.
MADCOF = -MFI8(dx, dxs, dans, 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, dans(35), slat, vbix(3),
  2 MADCOF)

30 VSAT = VBLMOD(dt, slat, slong, cd) ! unadjusted position
if(correc .eq. 'on') then
  if(dt1.lt.dt1 .or. dt1.gt.dt2) then ; recompute vector orbit
    dt1 = dt
    dt2 = dt + .02d0
    VOP = VUNIT4( VBLMOD(dt1, sstat1, slons1, cd1) )
    1 VBLMOD(dt2, sstat2, slons2, cd2) ; unit vector orbital plane
  end if

  adj = madc0f(1,1) + dx*(madc0f(2,1) + dx*madc0f(3,1))
  Angular adjustment which must be made to satellite position
  VEQ = VEC4( vop(1), vop(2), 0., )
  VX = VUNIT4( VEQ ** VOP ) * sign(1., elenm(3)-halfpi)
  VY = VOP ** VX ; the matrix MXR is now defined by equivalence
  MXR converts a vector from celestial to orbital plane coords
  adj = dxadj/(dx2 + 6.d0) ; weigths the adjustment
  rot(1,1) = cosine(adj)
  rot(2,1) = sine(adj)
  rot(1,2) = -rot(2,1)
  rot(2,2) = rot(1,1)
  VADJP0 = ROT * MTRAN4(MXR,3,3) * VSAT
  Adjusted position vector in the plane of the orbit
  VADJP = MXR * VADJP0 ; rotate back to celestial coords
else
  VSADJP = VSAT
end if

delast = dterop
return
end
VECTOR SATURNITE POSITION BY DROUWER-LYDDANE MODEL

VECTOR FUNCTION VBLMOD*(DTIME, FLAT,FLONG,CD)

SUBROUTINE VBLMOD(DTIME, FLAT,FLONG,CD, VEC)

*DTIME* IS THE TIME FOR WHICH A SATELLITE POSITION IS DESIRED,
EXPRESSED IN JULIAN DAY NUMBER, WHICH IS NOT, REPEAT, IS NOT
THE DAY OF THE YEAR. *DTIME* MAY BE OBTAINED FROM ORDINARY
TIME UNITS (YEAR,MONTH,DAY,HOUR,MINUTE,SECOND) USING THE
REAL*9 FUNCTION 'D2D1M'.

FOR A DEFINITION OF *JULIAN DAY NUMBER* SEE BOWDITCH,
"AMERICAN PRACTICAL NAVIGATOR", PUBLICATION NO. 9, VOL II,
P. 469, (DEFENSE MAPPING AGENCY)

THE RETURNED VALUES ARE GEODETIC LATITUDE, LONGITUDE,
CENTRAL DISTANCE (I.E. FROM CENTER OF EARTH TO SATELLITE,
IN KILOMETERS), AND THE CELESTIAL POSITION VECTOR. THE LATTER
IS RETURNED THRU THE FUNCTION NAME IF THIS ROUTINE IS CALLED
AS A "FUNCTION VALUE".

THE COMMON BLOCK /BLXTRA/ RETURNS THE B-L ELEMENTS UPDATED
TO THE GIVEN INPUT TIME.

THE ORBITAL PARAMETERS MUST ALREADY HAVE BEEN PUT INTO THE
COMMON BLOCK /VBL/ BY SOME OTHER ROUTINE, SUCH AS *VBLSET*
OR *GETHE*.

THIS ROUTINE USES THE SAME ORBITAL PREDICTION SOFTWARE USED BY
NOCIA/NEODIS, AND HENCE CANNOT FAIL TO BE UTTERLY CORRECT.

PARAMETER ( DTR = 6.283185307179586D+0/360.D+0 )

IMPLICIT REAL*8 (A-H,O-Z)
REAL*4 GIVENS,VLL(3),VLL(3),FLAT,FLONG,CD

DIMENSION BLLAST(6), OSCUL(6), AUX(5),
1 VECPTH(3)

COMMON/LCNST/ DTSEC,SCNDS,RADAR,GRG, DARG(7)
COMMON/VBL/ DTEPOC, GIVENS(6)
COMMON/BLXTRA/ DTPREP, DBLPRE(6), VECVEL(3)

DATA DTLAST, BLLAST/ -1.D+0, 6*O.D+0/, ANOM/-99999.0/
XXXXX XXXXXXX XXXXXXX XXXXXXX XXXXXXX XXXXXXX XXXXXXX XXXXXXX XXXXXXX

IF(DTIME.LT. DTEPOC) GO TO 900

DO 10 J = 1, 6
10 BLLAST(J) = GIVENS(J)

DTSECS = 86400.D+0 * (DTIME - DTEPOC)
IF(GIVENS(6).NE.ANOM) CALL BROLYD( OSCUL, BLLAST, 0, 1, 1, AUX)
RE-INITIALIZES PREDICTION SOFTWARE IF WE HAVE CHANGED ORBITAL
PARAMETERS SINCE THE PRECEDING CALL.

ANOM = GIVENS(6)
CALL BROLYD( OSCUL, BLLAST, 2, 2, 1, AUX)
BROLYD COMPUTES OSCULATING KEPLERIAN VALUES VALID AT TIME
DTIME FROM B-L ELEMENTS VALID AT EPOCH "DTEPOC".

RETURN UPDATED B-L ELEMS THRU COMMON /BLXTRA/

DO 20 J = 1, 6
20 DBLPRE(J) = BLLAST(J)

DTPREP = DTIME
EPOCH OF PREDICTED PARAMETERS

CALL CELEM(OSCUL, GRAV, VECPOS, VECVEL)
CELEM CONVERTS OSCULATING TO CELESTIAL

DO 30 J = 1, 3
30 VEC(J) = VECPOS(J)

CALL VCOORD( DTIME, VEC, 'CLL', VLL)
CD = CENDIS
FLAT = VLL(1)
FLONG = VLL(2)
RETURN

FLAT = -999999.
CD = 0.
VEC(1) = 0.
VEC(2) = 0.
VEC(3) = 0.
RETURN

END

C*********************************************************
C SUBROUTINE BROLYD
1 (OSCELE, DPHEL, IPERI, IPASS, IDMEEAN, OPREL)
C*********************************************************
C* REF.* ** BROUWER-LYDDANE ORBIT GENERATOR ROUTINE**
C* (X-553-70-223)
105 C* BY E.A. GALBREATH 1970
106 C* MODIFIED 7/31/74 VIONA BROWN AND R.A. GOPDON TO INTERFACE WITH GTDS*
107 C* FURTHER MODIFIED 11 DEC 85 BY F.W. MAGLE, NOAA/NE3DIS, TO
108 C* PARAMETERIZE THE FRACTIONAL CONSTANTS FOR GREATER SPEED*
110 C***********************************************************************
111 C C* IMPLICIT REAL*R(A-H,0-Z)
112 C***********************************************************************
113 C PARAMETER(
114 X F308=3.000/8.000,1,000,000
115 X F1D2=1.000/2.000,1,000,000
116 X F3D2=3.000/2.000,1,000,000
117 X F1D4=1.000/4.000,1,000,000
118 X F5D4=5.000/4.000,1,000,000
119 X F1D8=1.000/8.000,1,000,000
120 X F5D12=5.000/12.000,1,000,000
121 X F1D16=1.000/16.000,1,000,000
122 X F1D16=1.000/16.000,1,000,000
123 X F1D16=1.000/16.000,1,000,000
124 X F5D24=5.000/24.000,1,000,000
125 X F3D32=3.000/32.000,1,000,000
126 X F1D32=1.000/32.000,1,000,000
127 X F5D64=5.000/64.000,1,000,000
128 X F3D84=3.000/84.000,1,000,000
129 X F5D76=3.000/76.000,1,000,000
130 X F5D52=3.000/52.000,1,000,000
131 X F1D3=1.000/3.000,1,000,000
132 X F5D16=5.000/16.000,1,000,000)
133 C DIMENSION OSCELE(6),DPELE(6),ORBEL(5)
135 C DATA PMU, RE/1.0D+0, 1.0D+0, BKSUPC/0.0D+0/
136 C DATA PI2/6.283185307179586D+0/
137 C DATA (2**6) DPELE(1)/AE
138 C DATA (2**6) DPELE(2)/AE
139 C DATA (2**6) DPELE(3)/AE
140 C DATA (2**6) DPELE(4)/AE
141 C DATA (2**6) DPELE(5)/AE
142 C DATA (2**6) DPELE(6)/AE
143 C DATA (2**6) DPELE(7)/AE
144 C DATA (2**6) DPELE(8)/AE
145 C DATA (2**6) DPELE(9)/AE
146 C DATA (2**6) DPELE(10)/AE
147 C DATA (2**6) DPELE(11)/AE
148 C DATA (2**6) DPELE(12)/AE
149 C DATA (2**6) DPELE(13)/AE
150 C DATA (2**6) DPELE(14)/AE
151 C DATA (2**6) DPELE(15)/AE
152 C DATA (2**6) DPELE(16)/AE
153 C DATA (2**6) DPELE(17)/AE
154 C DATA (2**6) DPELE(18)/AE
155 C DATA (2**6) DPELE(19)/AE
156 C DATA (2**6) DPELE(20)/AE
157 C DATA (2**6) DPELE(21)/AE
158 C DATA (2**6) DPELE(22)/AE
159 C DATA (2**6) DPELE(23)/AE
160 C DATA (2**6) DPELE(24)/AE
161 C DATA (2**6) DPELE(25)/AE
162 C DATA (2**6) DPELE(26)/AE
163 C DATA (2**6) DPELE(27)/AE
164 C DATA (2**6) DPELE(28)/AE
165 C DATA (2**6) DPELE(29)/AE
166 C DATA (2**6) DPELE(30)/AE
167 C DATA (2**6) DPELE(31)/AE
168 C DATA (2**6) DPELE(32)/AE
169 C DATA (2**6) DPELE(33)/AE
170 C DATA (2**6) DPELE(34)/AE
171 C DATA (2**6) DPELE(35)/AE
172 C DATA (2**6) DPELE(36)/AE
173 C DATA (2**6) DPELE(37)/AE
174 C DATA (2**6) DPELE(38)/AE
175 C DATA (2**6) DPELE(39)/AE
176 C DATA (2**6) DPELE(40)/AE
177 C DATA (2**6) DPELE(41)/AE
178 C DATA (2**6) DPELE(42)/AE
179 C DATA (2**6) DPELE(43)/AE
180 C DATA (2**6) DPELE(44)/AE
181 C DATA (2**6) DPELE(45)/AE
182 C DATA (2**6) DPELE(46)/AE
183 C DATA (2**6) DPELE(47)/AE
184 C DATA (2**6) DPELE(48)/AE
185 C DATA (2**6) DPELE(49)/AE
186 C DATA (2**6) DPELE(50)/AE
187 C DATA (2**6) DPELE(51)/AE
188 C DATA (2**6) DPELE(52)/AE
189 C DATA (2**6) DPELE(53)/AE
190 C DATA (2**6) DPELE(54)/AE
191 C DATA (2**6) DPELE(55)/AE
192 C DATA (2**6) DPELE(56)/AE
193 C DATA (2**6) DPELE(57)/AE
194 C DATA (2**6) DPELE(58)/AE
195 C DATA (2**6) DPELE(59)/AE
196 C DATA (2**6) DPELE(60)/AE
197 C DATA (2**6) DPELE(61)/AE
198 C DATA (2**6) DPELE(62)/AE
199 C DATA (2**6) DPELE(63)/AE
200 C DATA (2**6) DPELE(64)/AE
201 C DATA (2**6) DPELE(65)/AE
202 C DATA (2**6) DPELE(66)/AE
203
157  HU = HDP
158  GO = GDP
159  BLP = BLDP
160  IFLG = 0
161  C---------------------------I
162  C COMPUTE MEAN MOTION I
163  C---------------------------I
164  ANU=DSQRT(BMU/A0**3)
165  C---------------------------I
166  C COMPUTE FRACTIONS I
167  C---------------------------I
168  BK2 = -F1D2*(BJ2*RE*RE)
169  BK3 = BJ3*RE**3
170  BK4 = F3DR*(BJ4*RE**4)
171  BK5 = BJ5*RE**5
172  GO TO 153
173  C
174  111  IF(IPERT.EQ.0)GO TO 7
175  C
176  IF(IDMEAN.NE.0) GO TO 202
177  C
178  ADP = DPELE(1)/AE
179  EDP = DPELE(2)
180  BIDP = DPELE(3)
181  HDP = DPELE(4)
182  GDP = DPELE(5)
183  BLD = DPELE(6)
184  C
185  153 EDP2=EDP*EDP
186  CN2 = 1.0 - EDP2
187  CN23 = CN2*CN2*CN2
188  CN = DSQRT(CN2)
189  GM2 = BK2/ADP**2
190  GMP2 = GM2/(CN2*CN2)
191  GM4 = BK4/ADP**4
192  GMP4 = GM4/CN**8
193  THETA = DCOS(BIDP)
194  THETA2 = THETA*THETA
195  THETA4 = THETA2*THETA2
196  C
197  202 IF(IDMEAN.EQ.0)GO TO 155
198  IF ( IPASS.EQ.2 ) GO TO 150
199  C---------------------------I
200  C COMPUTE LDOT, GDOT, HDOT I
201  C---------------------------I
202  157 BLDOT = CN*ANU*(GMP2*F3D2*(3.0*THETA2-1)+GMP2*F3D32*(THETA2
203  1*(-96.0*CN+30.0-50.0*CN2)*(16.0*CN+25.0*CN2-15.0)*THETA4
204  2*(144.0*CN+25.0*CN2+105.0)))+EDP2*GMP4*F15D16*(3.0*35.0*THETA4
205  3-30.0*THETA2))
206  GDOT = ANU*(F5D16*GMP4*(THETA2*(126.0*CN2-270.0)+THETA4*(385.0
207  1-129.0*CN2)-5.0*CN2+21.0)+GMP2*(F3D32*GMP2*(THETA4*(45.0*CN2
208  2+360.0*CN+385.0)+THETA2*(90.0-122.0*CN+126.3*CN2)+(24.0*CN

204
3 + (2.5 * CN2 - 35) + (F3D2 * (5 * THETA2 - 1))

HCT = ANUt * (GMP * F5D4 * THETA * (5.0 - GMP2) * (5.0 - CN2 - GMP2) * CN2)

1 * (GMP2 * F3D2 * (THETA2 * (12.0 * CN2 + 5.0 - CN2 - 5.0) - THETA2 * THETA2 * (5.0 - CN2)

2 * (5.0 * CN2 + 35.0)) - 3 * THETA2)

155 IF IFI=6 THEN GO TO 19

CI -------------------------------I

CI COMPUTE ISURC TO TEST CRITICAL INCLINATION I

CI -------------------------------I

HTSUBC= (1.0 - 5.0 * THETA2) * ((-2) * (5.0 * THETA4 * THETA2) * (GMP2 * EDP2))

IFGLG=1

CI -------------------------------I

CI FIRST CHECK FOR CRITICAL INCLINATION I

CI -------------------------------I

IF (ISUBC > GT.BKSUBC) GO TO 158

ASSIGN 163 TO ID8

GO TO 159

C -------------------------------I

C IS THERE CRITICAL INCLINATION I

C -------------------------------I

1.0 IF (ISUBC > GT.BKSUBC) GO TO 150

150 IF (IPERT.EQ.1) GO TO 150

GMP3 = BK3 / ADP ** 3

GMP3 = GM3 / (CN2 * CN2 * CN2)

GMP3 = GMP3 / CN3

GMP5 = GM5 / CN5

GMP5 = GMP5 / CN5

GMP5 = GMP5 / GM3

GMP2 = GMP2 / GMP3

GMP2 = GMP2 / GMP3

GMP2 = GMP2 / GM3

GMP2 = GMP2 / GM3

C -------------------------------I

CI COMPUTE A1 - AP I

C -------------------------------I

A1 = (F1D8 * GMP2 * CN2) * (1.0 - 11.0 * THETA2 - ((40.0 * THETA4) / (1.0 - 5.0 * THETA2

1.0))

A2 = (F5D12 * G4D2 * CN2) * (1.0 - ((8.0 * THETA4) / (1.0 - 5.0 * THETA2)) - 3.0

1.0 * THETA2)

A3 = G5D2 * (3.0 * EDP2 + 4.0)

A4 = G5D2 * (1.0 - (24.0 * THETA4) / (1.0 - 5.0 * THETA2) - 9.0 * THETA2)

A5 = (G5D2 * (3.0 * EDP2 + 4.0)) * (1.0 - (24.0 * THETA4) / (1.0 - 5.0 * THETA2) - 9.0

1.0 * THETA2)

A6 = G3D2 * F1D4

SINH = SINH(BIDP)

A10 = CN2 * SINI

A7 = A6 + A10

A8 = G5D2 * EDP * (1.0 - (16.0 * THETA4) / (1.0 - 5.0 * THETA2) - 5.0 * THETA2)

A9 = A8 + EDP

C -------------------------------I

CI COMPUTE B13-B1E

C -------------------------------I

B13 = EDP * (A1 - A2)

B14 = A7 * F5D6 * A5 * A10

B15 = A8 * A10 * F35384
261  C
262  C       COMPUTE A11-A27
263  C
264  A11=2.0*EDP2
265  A12=3.0*EDP2+2.0
266  A13=THETA2*A12
267  A14=(5.0*EDP2+2.0)*(THETA4/(1.0-5.0*THETA2))
268  A17=THETA4/((1.0-5.0*THETA2)*(1.0-5.0*THETA2))
269  A15=(EDP2*THETA4*THETA2)/((1.0-5.0*THETA2)*(1.0-5.0*THETA2))
270  A16=THETA2/((1.0-5.0*THETA2))
271  A1#=EDP+SINI
272  A19=A18/(1.0+CN)
273  A21=EDP+THETA
274  A22=EDP2*THETA
275  SINI2=DSIN(BIDP/2.0)
276  COSI2=DCOS(BIDP/2.0)
277  TAN12=DTAN(BIDP/2.0)
278  A26=16.0*A16+40.0*A17+3.0
279  A27=A22+F108*(11.0+200.0)*A17+80.0*A16
280  CI---------------
281  CI COMPUTE B1-B12 I
282  CI---------------
283  B1=CN*(A1-A2)-((A11-40.0*A15-40.0*A14-11.0*A13)*F1016*(11.0+200.0)
284  1*A17+80.0*A16)*A22+F108)*GMP2+((-80.0*A15-A10*A14-3.0*A13)*A11)
285  2+F5D24+F5D12*A26+A22)*G40G2
286  B2=A6*A19*(2.0+CN-EDP2)+F5D64*A5*A19*CN2-F15D32*A4*A18*CN*CN2
287  B7=(F5D64*A5*A6)*A21*TAN12+(9.0*EDP2+26.0)*F5D64*A4*A18+F15D32*A3
288  2*A21*A26*SINI+(1.0-THETA)
289  B3=((8.0+0*A17+5.0+32.0*A16)*A22*SINI*(THETA=1.0)*F35576*G5DG2*EDP)
290  1-((A22*TAN12+2.0*EDP2+3.0*(1.0-CN2-CN2))*SINI)*F3552*ABP)
291  B4=CN*EDP*(A1-A2)
292  B5=((9.0*EDP2+4.0)*A10*A4*F5D64*A7)*CN
293  B6=F35364*A8*CN2*CN*SINI
294  B7=((CN2*A18)/(1.0-5.0*THETA2))*(F108*GMP2*(1.0-15.0*THETA2)+(1.0
295  1-7.0*THETA2)*G40G2*(-F5D12))
296  B8=F5D64*(A3*CN2*(1.0-9.0*THETA2-(24.0*THETA4/(1.0-5.0*THETA2))))
297  1*A6*CN2
298  B9=8*F35384*CN2
299  B10=SINI*(A22*A26*G4DG2*F5D12-A27*GMP2)
300  B11=A21*(A5+F5D64*A6*A3*A26*F15D32*SINI*SINI)
301  B12=-(8.0*A17+32.0*A16+5.0)*((A22*EDP*SINI*SINI*F35576*G5DG2)+(A8
302  1*A21+F3552))
303  150 IF(IPERT=EQ.0)GO TO 7
304  IF(IDMEAN=EQ.0)GO TO 4
305  C-----------------------------I
306  C COMPUTE SECULAR TERMS I
307  C-----------------------------I
308  CI-----------------------------I
309  CI **MEAN** MEAN ANOMALY I
310  CI-----------------------------I
311  BLDP = ANU*DELT + BLD*DELT + BLO
312  BLDP = DMO(BLP1,PI2)
IF(BLDP.LT.0.0D0)BLDP = BLDP + PI2
CI MEAN ARGUMENT OF PERIGEE I
CI----------------------------------------I
GDP = GDSP*DELT + GO
GDP = D*MOD(GDP,PI2)
IF(GDP.LT.0.0D0)GDP = GDP + PI2
C MEAN LONGITUDE OF ASCENDING NODE
HDP = HDT*DELT + HO
HDP = D*MOD(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
C CI COMPUTE TRUE ANOMALY(DOUBLE PRIMED) I
C CI----------------------------------------I
EADP = DKEPLR(BLDP,EDP)
SINDE = DSIN(EADP)
COSDE = DCOS(EADP)
SINFSD = CN*SINDE
COSFS = COSDE - EDP
FDP = DATANO(SINFSD,COSFS)
IF(IPERT.EQ.1) GO TO 7
C
C
DADR = (1.0 - EDP*COSDE)**(-1)
DADP = 1.0 + (1.0 - EDP*COSDE)
SINFSD = SINFSD*DADR
COSFS = COSFS*DADR
CS2GFD = DCOS(2.0*GDP+2.0*FDP)
DADR2 = DADR*DADR
DADR3 = DADR2*DADR
COSFD2 = COSFD*COSFD
C CI----------------------------------------I
C CI COMPUTE A(SEMI-MAJOR AXIS) I
C CI----------------------------------------I
A = ADP*(1.0 + G2*M*(3.0*THETA2-1.0) + (EDP2/CN23) + (CN+(1.0/(1.0 + CN)))*(3.0*THETA2-1.0) + (EDP*COSFD)*(3.0*3.0*EDP + 2.0*COSFD*EDP + COSFD2) + 3.0*(1.0-THETA2) + DADR3*CS2GFD))
SN2GFD = DSIN(2.0*GDP+2.0*FDP)
SNF2G = DSIN(2.0*GDP+FDP)
CS2GFD = DCOS(2.0*GDP+FDP)
SN2G = DSIN(2.0*GDP)
CS2G = DCOS(2.0*GDP)
SN3G = DSIN(3.0*GDP)
365  CS3GD=DGOS(3.0*GDP)
366  SN3FGD=DSIN(3.0*DF*F+2.0*GDP)
367  CS3F9GD=DCOS(3.0*DF*F+2.0*GDP)
368  SINGD=DSIN(GDP)
369  CSGD=DCOS(GDP)
370  GO TO IDR*(163,164)
371  163 DLT1C=B14*SINGD+B13*CS2GD-B15*SN3GD
372  CI-------------------------------I
373  CI COMPUTE (L+G+H) PRIMED I
374  CI-------------------------------I
375  BLGH=HDP+GDP+BLDP+B3*CS3GD+B1*SN2GD+B2*COSGD
376  BLGH=MOD(BLGH,P12)
377  IF (SINH,P.LT.0.000) BLGH=BLGH+PI2
378  DPDL=E4*SN2GD-R5*COSGD+B5*CS3GD-F1D4*CN2*CN*GMP*(2.0*(3.0*THETA2)
379  1-1.0)*(DADR2+CN2+DADR+1.0)*SINFD+3.0*(1.0-THETA2)*((-DNDR2*CN2
380  2+DADR+1.0)*SN2GD+(DADR2*CN2+ADRR+1.0)*DN2GD))
381  DLTI=F1D2*THETA*GMEM+2*SN3GD)
382  1-(S2/P14CN2)*(B8*SING+D7*CS2GD-B9*SN3GD)
383  SINGH=(1.0/CS2GD)*((F1D2*CN2*CS3GD+SIGDS+B10*SN2GD-(F1D2*GMP2
384  1*THETA+SN2GD*(6.0*(EDP+SN2GD-FLDP-FDP)-(3.0*(SN2GD+EDP+SN2GD+EDP
385  2+SN3GD)))))
386  CI-------------------------------I
387  CI COMPUTE (L+G+H) I
388  CI-------------------------------I
389  164 BLGH=BLGH+(1.0*(CN+1.0)))+F104*EDP*GMP2*CN2*(3.0*(1.0-THETA2)
390  1*(SN3F9GD*F1D3+DNDR2*CN2+DNDR)+SNF2GD*(1.0*(DNDR2*CN2+DNDR))+2.0*
391  2*SINFD*(3.0*THETA2-1.0)*(DNDR2*CN2+DNDR+1.0)))+GMP2*F3D2*(-2.0*
392  3*THETA1.0*5.0*THETA2)*EDP+SN3gd*EDP+BLDP))+(3.0*2.0*THETA-5.0*
393  4*THETA2)*GMP2*F1D4*(3.0*(SN2GD+EDP+SN2GD))
394  BLGH=MOD(BLGH,F12)
395  IF (BLC+LT.0.000) BLGH=BLGH+PI2
396  DLTI=DLTI*(F1D2*CN2+(3.0*(1.0/CS3GD))*GMP1*(1.0-THETA2)
397  1*CS2GD*(3.0*EDP+CS2GD+3.0+CS3GD+EDP+CS2GD+EDP)-GMP2
398  2*(1.0-THETA2)*(3.0*CS2GD+CS3GD))+(3.0*THETA2-1.0)*GMP2*(1.0/
399  3*(CN*3)*EDP*(1.0+CN)+3.0*EDP+CS2GD+3.0*CS2GD+*
400  4*EDP+CS2GD)*CS2GD)
401  EDPLD=EDPD+EDPD
402  EDPD*,=EDP+DLEE*(EDP+DLT)
403  CI-------------------------------I
404  CI COMPUTE F(ECCENTRICITY) I
405  CI-------------------------------I
406  E=DSRT(EDPLD*EDPTCD)
407  SINDH=DSINDH*DSINH
408  SQR=DSRT(DLTI*CS12*F1D2+SN2I2)*(DLTI*CS12*F1D2*SN2I2)
409  DSRT=DSRT(SINDH*SN2I2)
410  CI-------------------------------I
411  CI COMPUTE BI (INCLINATION) I
412  CI-------------------------------I
413  BI=DADSIN(SQRI)
414  BI=2.0*5
415  BI=MOD(BI,P12)
416  IF (PI+LT.0.004) BI=BI+P12

208
CI------------I
CI CHECK FOR E (ECCENTRICITY) = 0 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 = DDATA0(ARG1, ARG2)
C
        OSCELE(1) = A * AE
        OSCELE(2) = E
        OSCELE(3) = BI
        OSCELE(4) = H
        OSCELE(5) = G
        OSCELE(6) = BL
C
473 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 DLTI = 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.0.00D0)BL=BL+PI2
GO TO 145

CI------------------------I
CI COMPUTE H(LONGITUDE OF ASCENDING NODE) I
CI------------------------I
169 ARG1=SINDH*COSHDP+INHDP*(F102+DLTI*COSI2+SI2)
ARG2=COSHDP*(F102+DLTI*COSI2+SI2)-(SINDH*INHDP)
H=DATAN2(ARG1, ARG2)
H=DMOD(H, PI2)
IF(H<LT.0.00D0)H=H+PI2
GO TO 146

45 CONTINUE
RETURN
END

C RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
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
DOOS SYSTEM DESCRIPTION, SUBROUTINE KEPLR1
F. EXCORAL-METHODS OF ORBIT DETERMINATION
X-552-67-421, COMPARISON FOR ITERATIVE TECHNIQUES FOR THE
SOLUTION OF KEPLER'S EQUATION, I. COLE AND R. BORCHERS
CHARLES K. CAPPERS, CODE 553.2, GSFC
IMPLICIT REAL*8(A-H,O-Z)
DATA MAX /10/
DIMENSION PV(3),VV(3),ORBEL(6)
DATA TOL /+0.5D-16/
ITER = 0
IF THIS IS ELLIPTIC OR HYPERBOLIC ORBIT
IF (ORBEL(1)LE.0.000 AND ORBEL(2)GT.1.000) GO TO 50
ELLIPSE ORBIT TAKES THIS ROUTE
FIRST FIND ELLIPSE ANOMALY VIA NEWTONS (MILES STANDISH VERSION)
E1 = ORBEL(6)
10 F = E1 - (ORBEL(2)*DSIN(E1) - ORBEL(6))
D = 1.000 - (ORBEL(2)*DCOS(E1 - 0.500*D))
E2 = E1 - (F/D)
IF (DABS(E1-E2)TOL)40,40,20
ITER = ITER + 1
E1 = E2
IF (ITER MAX) 10,10,30
LECEN ANOMALY CONVERGED, NOW GET X0,Y0,R
30 NERR = 13
40 COSE = DCOS(E2)
SINE = DSIN(E2)
TEMP = 1.000 - ORBEL(2) - ORBEL(2)
X0 = ORBEL(1) * (COSE - ORBEL(2))
Y0 = ORBEL(1) * (DSORT(TEMP)*SINE)
R = ORBEL(1) * (1.000 - ORBEL(2) - COSE)
X0D = (-DSORT(GMC*ORBEL(1))*SINE)/R
Y0D = (-DSORT(GMC*ORBEL(1)*(TEMP))*COSE)/R
GO TO 10

HYPERS ORBITS TAKE THIS ROUTE
50 E1 = ORBEL(6) / 2.000
60 F = ORBEL(2)*DSINH(E1) - E1 - ORBEL(6)
D = ORBEL(2)*DCOSH(E1 - 0.500*D) - 1.000
E2 = E1 - (F/D)
IF (DABS(E1-E2)TOL)90,90,70
70 ITER = ITER + 1
E1 = E2
IF (ITER MAX) 60,60,90
SET UP ERROR CODE FOR NON-CONVERGENCE PRIOR TO EXIT
80 NERR = 14
ECCENTRIC ANOMALY COMPUTE, NOW GET X0,Y0,R

211
90 COSF = DCOSH (E2)
91 SINF = DSINH(E2)
92 TEMP = ORBEL(2) + ORBEL(2) - 1.0D0
93 XG = ORBEL(1)*(COSF - ORBEL(2))
94 YO = - ORBEL (1)*DSQRT (TEMP) * SINF
95 R = ORBEL(1)*(1.0D0 - ORBEL(2) * COSF)
96 XOD = (-DSQRT(-GMC*ORBEL(1))*SINE)/R
97 YOD = (DSQRT(-GMC*ORBEL(1)*TEMP)*COSF) / R
98 100 COSO = DCOS (ORBEL(5))
99 SINO = DSIN (ORBEL(5))
100 COSOM = DCOS (ORBEL(4))
101 SINOM = DSIN (ORBEL(4))
102 COSI = DCOS (ORBEL(3))
103 SINI = DSIN (ORBEL(3))
104 BI1 = COSO * COSOM - SINO * SINOM * COSI
105 B12 = -SINO * COSOM - COSO * SINOM * COSI
106 B21 = COSO * SINOM + SINO * COSOM * COSI
107 B31 = SINO * SINI
108 B32 = -SINO * COSOM - COSO * SINOM * COSI
109 B22 = -SINO * SINOM + COSO * COSOM * COSI
110 B32 = COSO * SINI
111 C NOW MULTIPLY 3 X 2 MATRIX BY 2 X 1 VECTORS FOR POSITION, VELOCITY.
112 PV(1) = BI1 * X0 + B12 * Y0
113 PV(2) = B21 * X0 + B22 * Y0
114 PV(3) = B31 * X0 + B32 * Y0
115 VV(1) = BI1*XOD + B12 * YOD
116 VV(2) = B21 * XOD + B22 * YOD
117 VV(3) = B31 * XOD + B32 * YOD
118 999 RETURN
119 END
120 C RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
121 C BLOCK DATA
122 C IMPLICIT REAL*8 (A-H,O-Z)
123 C NAME = BLCNST
124 C LANGUAGE = FORTHXP TYPE = PROGRAM
125 C THIS COMMON BLOCK WAS UPDATED MARCH 28, 1984 TO INCLUDE XKE AND ESQ
126 C BY E. HARROD S/SP12
127 C THIS BLOCK DATA IS COMPILRED WITH THE ROUTINE PSCEAR, ANY PROGRAM
128 C USING PSCEAR DOES NOT NEED TO RECOMPILE THIS BLOCK DATA
129 C*******************************************************************************
130 C*******************************************************************************
132 C*******************************************************************************
133 DATA TTO,R,GM,AE,BJ2,BJ3,BJ4,BJ5,FLTINV,XKE,ESQ/2.0D0,
134 398600.80D0,6378.13500,-7.10826158D-02,0.25388100D-05,
135 0.16559700D-05,0.21848266D-06,0.298.250D0,0.74366916D-01,
136 0.5994317778266721D-02/
C REAL FUNCTION DATAN08 (DA, DB)
C IMPLICIT REAL*8 (A-H, O-Z)
C DATA PI2/6.28318530717958650/0/
C DA = DATAN2 (DA, DB)
C IF (DA ELT. 0.0) DA = DA + PI2
C DATANC = DA
C RETURN
C END
C C FUNCTION DKEPLR (M, E)
C IMPLICIT REAL*8 (A-H, O-Z)
C REAL*8 M, PI2/6.28318530717958650, TOL/0.5D-15/
C C SUBROUTINE TO SOLVE KEPLER'S EQ.
C 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
C IF (M) 1, 2
1 EA = M + E * DSIN (M)
D0 22 I = 1, 12
C OLDEA = EA
C FE = EA - E * DSIN (EA) - M
C EA = EA - FE / (1 - E * DCOS (EA - 0.5D0 * FE))
C TEST FOR CONVERGENCE
C DELEA = DABS (EA - OLDEA)
C IF (DELEA LE TOL) GO TO 2
C 22 CONTINUE
C 2 EA = DMOD (EA, PI2)
C DKEPLR = EA
C RETURN
C END
C /*
C //EDIT*SYSPRINT DD SYSOUT=A
C //EDIT*SYSIN DD *
C NAME VBLMOD(R)
C /*
C //