RADIATIVE INPUT to a HIGH ALTITUDE OBJECT CALCULATED USING GOES IR DATA

Radiative Input to a High Altitude Object Calculated Using GOES IR Data

Prepared by

Space Science and Engineering Center
University of Wisconsin-Madison
Madison, Wisconsin 53706

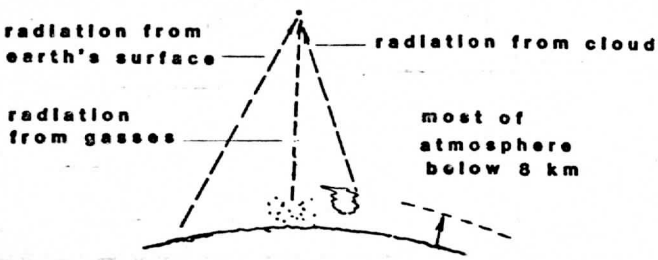
March 1982

Barry B. Hinton

INTRODUCTION

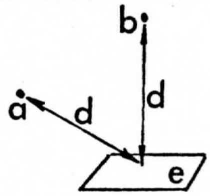
a. Idealized Problem

The objective is to compute the thermal radiation input to a body at high altitudes above the earth. For technical reasons the method delveloped can be validly applied when the altitudes are much greater than eight kilometers. For an object at such altitudes essentially all thermal radiation due to the atmosphere (including both clouds and radiating gasses such as water vapor and carbon dioxide) comes from below. Certainly for altitudes, as considered here, that exceed 80 km it is a very good approximation to imagine that all the thermal radiative input originates from a surface below the object. Thus, a single surface incorporates the radiative properties of both the atmosphere and the earth's surface.



To obtain the total input the earth below is divided into many small patches, the strength of the radiation emanating from each determined (per unit area of the patch). The product of the radiating strength and area of each of these patches (or area elements) is then added--if it can be seen from the test object or body. In this summation weight is also given to the distance from the emmitter and test body according to the "inverse square law".

Account must also be taken of the relative look angles between the test body and the area element ("cosine law") since a flat emitter viewed edgewise transfers less energy to a second object than the same emitter viewed straight on from the same distance. That is, a receives less than b from e.



In order to divide the surface below into the small patches an image from GOES is used.* That is, the pixels as defined by the GOES 10.5 micron IR channel (or multiples of them) are the elements of area we have used. The digital count value of each pixel is transformed to a brightness temperature (at 10.5μm), and the brightness temperature used with an empirical model developed by Smith et al. to obtain the strength of the radiation across the entire spectrum.**

---

* See the GOES/SMS User's Guide, Corbell, R. P., C. J. Callahan, W. J. Kotch, eds. Published jointly by NOAA (NESS) and NASA, 1976.

** Smith, W. L., L. D. Herman, T. Schreiner, H. B. Howell and P. Menzel, Radiation Budget Characteristics of the Onset of the Summer Monsoon, in International Conference, on Early Results of FGGE and Large Scale Aspects of Its Monsoon Experiments. WMO, Geneva 1981.

b. Results

We have presented below in Table 1 the results which were obtained using the method discussed in more detail below. In a later section these results are also shown in graphical form. Note that for t>1359 the results are given at 10 second intervals.

Table 1
Results for Trajectory Provided

Time(s)[1]	Flux(Wm^{-2})	Time(s)	Flux(Wm^{-2})	Time(s)	Flux(Wm^{-2})
1159	207.0	1309	216.5	1399	211.0
1169	207.7	1319	217.8	1409	208.6
1179	208.6	1329	218.7	1419	205.8
1189	209.5	1339	218.0	1429	202.2
1199	210.5	1349	216.7	1439	197.8
1209	211.3	1359	217.4	1449	193.1
1229	213.0	1369	215.4	1459	188.6
1259	214.5	1379	214.3	1469	185.6
1279	216.3	1389	213.9	1479	185.3

[1] 0.3660 should be added to every time entry

The word flux is used for power per unit area rather than a more precise, but cumbersome, term such as flux density.

Mathematical Description

In this section we define the geometrical quantitites and derive the expressions required to program the algorithm outlined in the previous section. Matters pertaining to radiative quantities (e.g. emissivity, Plank function, etc.) are referred to the books of Liou and Fleagele and Businger.* Following this is a listing of the computing program. Parts of the program are not easy to understand, because much use is made of the McIDAS system subroutines and related terminology. Documentation for these is unpublished material obtainable from the University of Wisconsin Space Science and Engineering Center program librarian.

Figure 1 describes the transformation from latitude, longitude and altitude coordinates to a cartesian system (X, Y, Z). Figure 2 is intended to illustrate the general relationship of the test body (located at the "observation point") to a patch (or pixel) of area dA.

Let $\vec{r}_p$ be the vector OP. The earth centered cartesian coordinates are given by $\vec{r}_p = (X_p, Y_p, Z_p)$ where,

$$X_p = (a + \zeta)\ (\cos\theta_p\ \cos\phi_p)$$

$$Y_p = (a + \zeta)\ (\cos\theta_p\ \sin\phi_p)$$

$$Z_p = (a + \zeta)\ \sin\theta_p$$

The energy which crosses dA in time dt and which is confined to the solid angle $d\,\Omega$ (see Fig. 3) is:

$$dE = I_\lambda \cos\theta\, dA\, d\Omega\, dt\ d\lambda \qquad (1)$$

where I_λ is the intensity. If A emitts istropically, we may take

---

* Liou, K-N, An Introduction to Atmospheric Radiation, Academic Press, New York, 1980.
Fleagele, R. G. and J. A. Businger, An Introduction to Atmospheric Physics, (2nd ed.) Academic Press, New York, 1980.

** McIDAS stands for Man-Computer Interactive Data Access System.

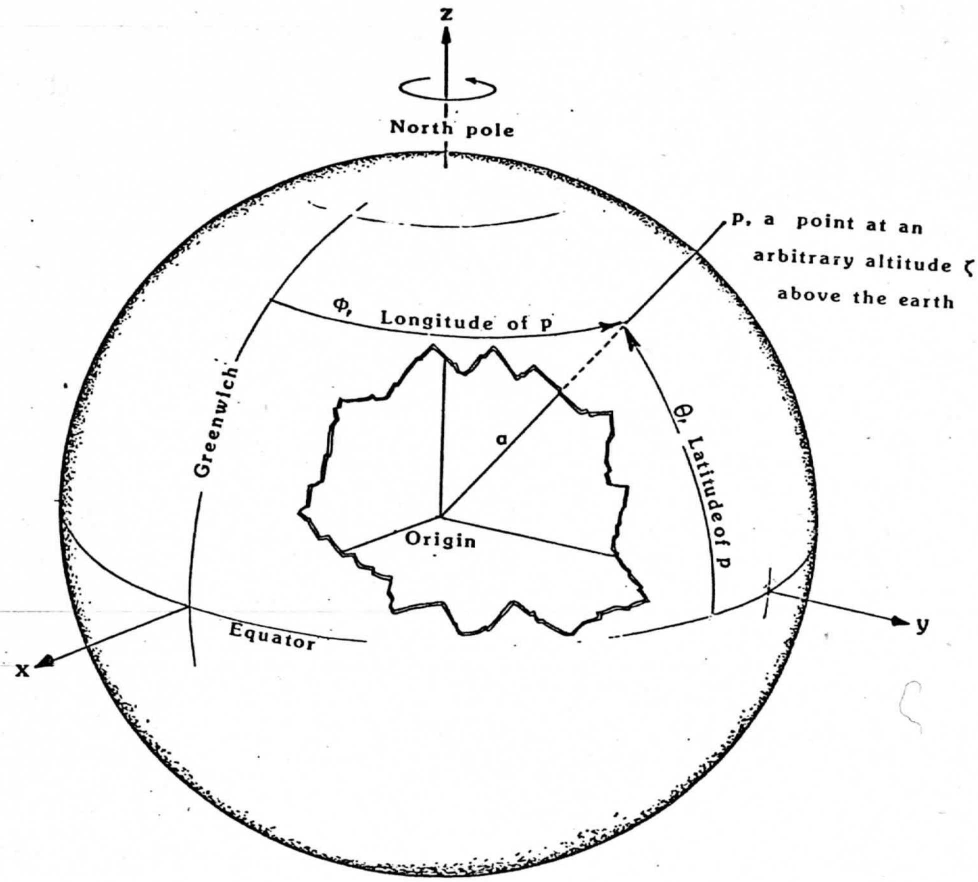


Figure 1. Earth centered cartesian coordinate system. An arbitrary point, p, is shown at an altitude ζ above the earth of radius a.

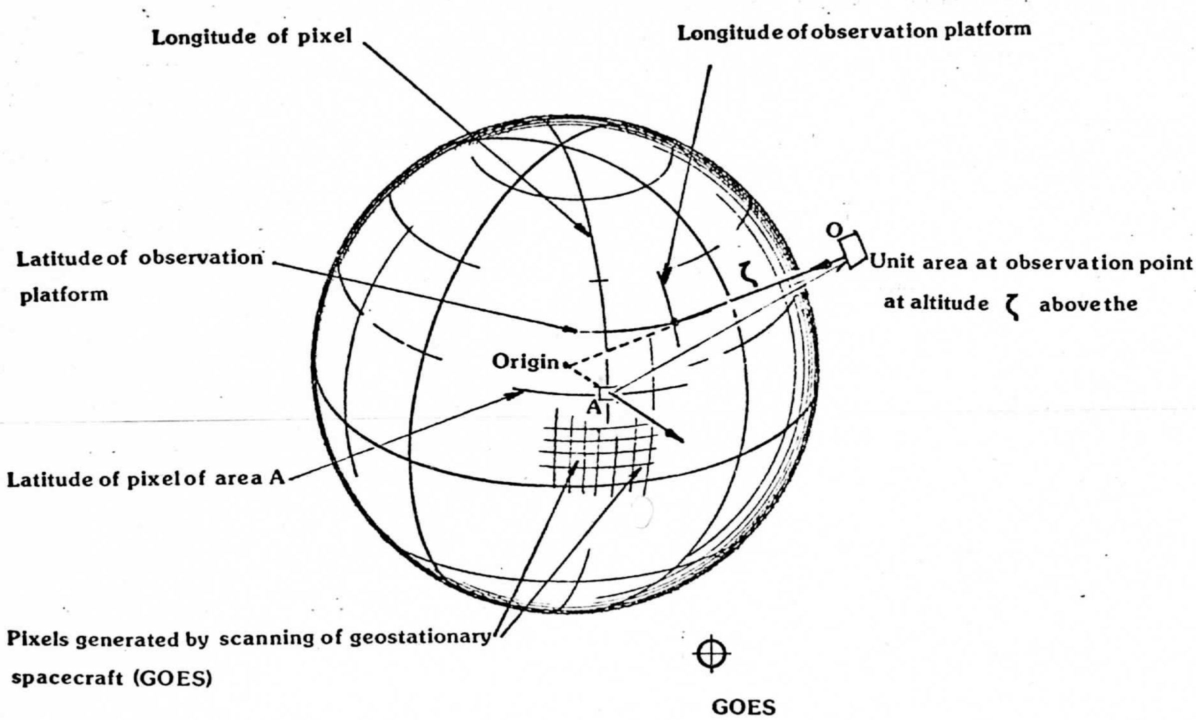


Figure 2. General relationships of GOES, observation point (0) and a patch on the surface (dA).

$$I_\lambda = \varepsilon_\lambda B_\lambda (T) \tag{2}$$

Where is the emissivity and B is the radiant intensity, or Plank, function of a black body. Thus, if $\varepsilon B = \int \varepsilon_\lambda B_\lambda (\lambda) d\lambda$ (1) becomes,

$$dE = \varepsilon B \cos\theta \; dA \; d\Omega \; dt \tag{3}$$

In this equation E is an 'effective' or mean emissivity.

The energy indident on dA_0 from dA is the intensity emitted by dA within the solid angle $d\Omega_o$ times dA_o and dt as well as $d\Omega$ itself See Figure 4.

$$dE = \varepsilon B\cos\theta \; dA \; dt \; d\Omega_o \tag{3^1}$$

However, in this case, $d\Omega_o$ can also be represented as,

$$d\Omega_o = dA_o \cos\theta_o / r^2_{Ao}$$

where r_{Ao} is the distance from dA to dA_o. Consequently

$$\frac{dE}{dt} = \frac{dA\cos\theta \; dAo\cos\theta o \; \varepsilon B}{r^2_{Ao}} \tag{4}$$

Actually, it is best to recast this equation in a vector notation. Let $\vec{r}_A$ be the position vector of the element dA and $\vec{r}_o$ that of dA_o

Also, let $\vec{r}_{Ao}$ be a vector from dA to dA_o, and $r_{oA} = -\vec{r}_{ao}$ a vector from dA_o to dA. We may therefore write (4) as,

$$\frac{dE}{dt} = \frac{dA(\hat{n}\cdot\vec{r}_{Ao})\; dAo(\hat{n}_o\cdot\vec{r}_{oA})\,\varepsilon B}{(r_{Ao})^4} \tag{5}$$

or,

$$\frac{dE}{dt} = \frac{dA(\hat{n}_A\cdot\; \vec{r}_{Ao})dAo(\hat{n}_o\cdot\vec{r}_{oA})\varepsilon B}{(r_{Ao})^2} \tag{5'}$$

As can be seen in Figures 3 and 4, n_A is the unit normal to dA and the unit normal to dA , $\hat{r}_{Ao}$ is $\vec{r}_{Ao}/|\vec{r}_{Ao}|$ and $\hat{r}_{oA} = \vec{r}_{oA}/|\vec{r}_{oA}|$.

In the program at hand dA is a unit area, i.e dA = 1, dA is the variable AREA, (XO, YO, Z) = $\vec{r}_o$ (XA, YA, ZA) = $\vec{r}_A$, $\vec{r}_{Ao}$ = (XA),YAO,ZAO), $\hat{r}_{Ao}$ = (XUAO,YUAO,ZUAO), $\hat{r}_{oA}$ = (XUOA,YUOA, ZUOA)

In this particular application, both dA_o and dA are considered horizontal hence, $\hat{n}_o = -\hat{r}_o = -\vec{r}_o/|\vec{r}_o|$ and $\hat{n}_A = \hat{r}_A = \vec{r}_A/|\vec{r}_A|$

Finally, the quantity εB, which is to be deduced from the GOES IR image is rendered by the equivalent expression in terms of flux density (F, (in Watt m^{-2}),

$$F/\pi = \varepsilon B \tag{6}$$

According to the empirical study of Smith <u>et al</u> cited above.

$$F = 0.543 \quad T^4_{GOES} + 44.538 \tag{7}$$

T_{GOES} is the GOES window IR temperature and the Stephan Boltzman constant = $5.66\ 10^{-8} W/M^2\ deg^{-4}$.

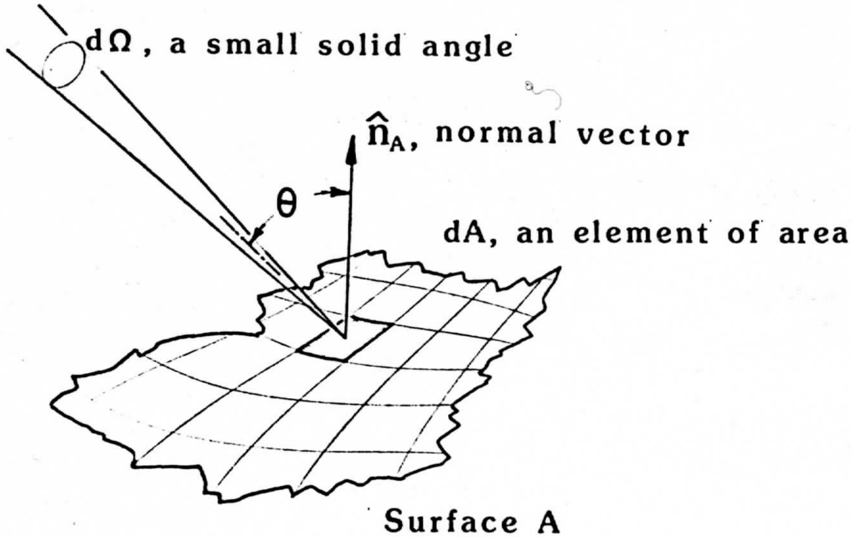


Figure 3. Terms relating to an emitting surface A.

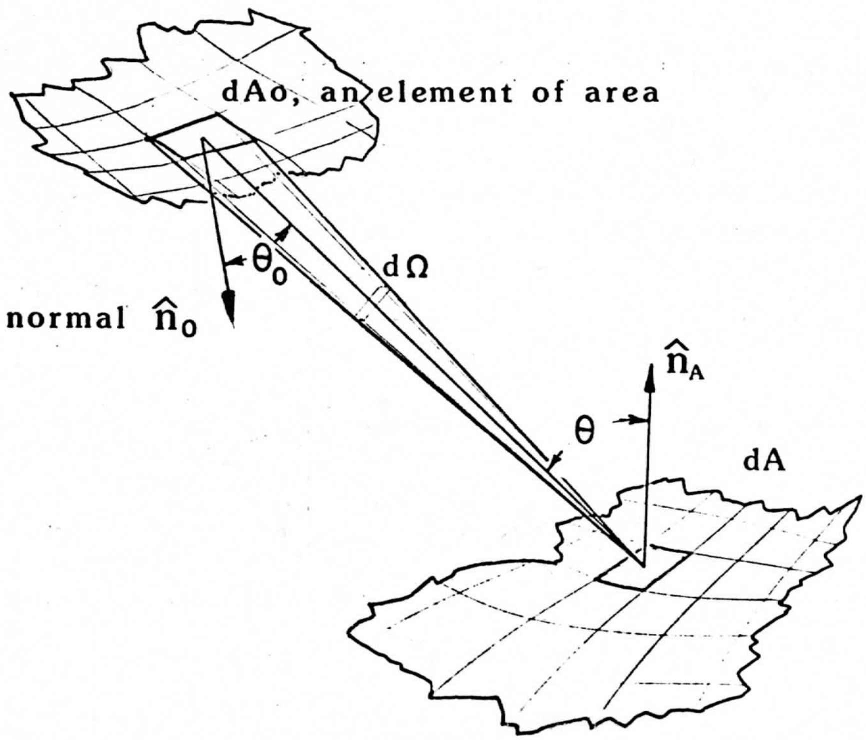


Figure 4. Radiant energy flux incident on an area dA_o.

If CT is a GOES digital count value T_{GOES} is given by (8),

$$T = \begin{cases} 330 - CT/2, & CT<176 \\ 418 - CT, & CT>176 \end{cases} \quad (8)$$

To obtain the total power incident on dA_o, one must sum over all the dA which can be seen. The dA are taken to be IR pixels in the GOES image. In order to be seen the pixel locations must satisfy the two conditions below.

Cond. 1: $\hat{n}_o \cdot \hat{r}_{oA} > 0$ (9)

If this condition is satisfied the look direction is downward, as shown in Figure 5.

Cond. 2: $\hat{n}_A \cdot \hat{r}_{Ao} > 0$ (10)

If this condition is not satisfied, it means that the radiation intercepts the earth as shown in Figure 6.

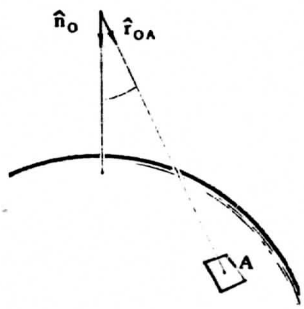


Figure 5. Illustration for condition 1.

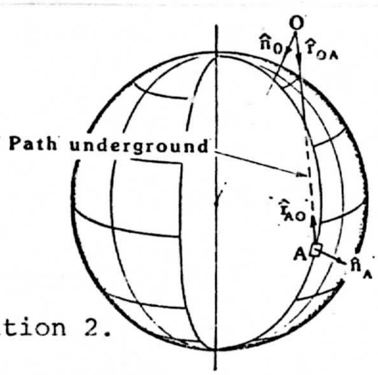


Figure 6.

Illustration for condition 2.

Program Listing

```
$JOB X U1000
$FORTRAN
      SUBROUTINE MAIN
C
C +---------------------------------------------------------------------+
C ! -->NAME                                                             !
C ! GTFLUX                                                              !
C ! -->SYNOPSIS                                                         !
C ! COMPUTE TEMPERATURE AT OBSERVATION POINT.                           !
C ! -->DESCRIPTION                                                      !
C ! FLUX IS INTEGRATED OVER A GIVEN AREA, "AS VIEWED" BY AN             !
C ! OBSERVATION POINT. RADIATIVE FLUX PER UNIT AREA IS WEIGHED.         !
C ! THERE ARE SIX INPUT PARAMETRES:                                     !
C !   ISTART  = MIN(1) = SERIAL NUMBER OF FIRST AREA.                   !
C !   NUMOFA  = MIN(2) = TOTAL NUMBER OF AREAS.                         !
C !   LAT     = MIN(3) = LATITUDE OF OBSERVATION POINT.                 !
C !   LON     = MIN(4) = LONGITUDE OF OBSERVATION POINT.                !
C !   IALTI   = MIN(5) = ALTITUDE OF OBSERVATION POINT.                 !
C !   NEW     = MIN(6) = ACTUAL NUMBER OF PIXELS/LINE AS OPPOSED TO     !
C !                      THOSE DISPLAYED BY THE LA COMMAND. THIS        !
C !                      MAY BE PROCURED WITH THE T KEY AND FROM        !
C !                      THE RASTOR SCREEN.                             !
C +---------------------------------------------------------------------+
C
      DIMENSION MIN(10),INBUF(840),IOUTBF(2520)
      DIMENSION NOUT(24),IGRAB(24)
      LOGICAL TRACE
      LOGICAL CRTRCE
      COMMON /NAVNUM/ PTIME,BETAIN,BETDOT,ATFRAC,ITYPE,INAV
      COMMON /COMP/ XO,YO,ZO,XA,YA,ZA
      COMMON /MAINIT/ NL,NE,ILIN,IELE,IDAY,JTIME,LINE,IEL
      COMMON /IRESOL/ LINRES,IELRES
      DATA TRACE /.TRUE./
      DATA CRTRCE /.TRUE./
      DATA A /6372.10E03/
      DATA XKM2ME /1000.0/
      DATA LNOFST,IEOFST /1,1/
      DATA MIN /'GTFLUX',8*0/
      CALL IQ(MIN)
      CALL OPN(6)
      ISTART = MIN(1)
      NUMOFA = MIN(2)
      LAT = MIN(3)
      LON = MIN(4)
      IALTI = MIN(5)
      NEW = MIN(6)
C
      IF (TRACE) PRINT 37
 37   FORMAT(1X,'-->COMMAND PARAMETERS:')
      IF (TRACE) PRINT 38,ISTART,NUMOFA,LAT,LON,IALTI,NEW
 38   FORMAT(1X,'ISTART=',I2,' NUMOFA=',I1,' LAT=',I7,
     *' LON=',I8,' IALTI=',I4,' NEW=',I3)
```

```
C
      DIST = FLOAT(IALTI) * XKM2ME
      IEND = ISTART + NUMOFA - 1
C
C +-------------------------------------------------------------------+
C ! IN FUTURE, IF MORE THAN ONE AREA IS USED THEN THIS LOOP MAY      !
C ! BE USED. SLIGHT MODIFICATION WILL BE IN ORDER. LIKE SATELITE     !
C ! POSITIONS WOULD BE READ OFF A FILE INSTEAD OF BEING KEYED IN.    !
C +-------------------------------------------------------------------+
C
      DO 500 IA = ISTART,IEND
C
C +-------------------------------------------------------------------+
C ! TRANSFORM SATELITE POSITION.                                      !
C +-------------------------------------------------------------------+
C
      CALL CARTES(LAT,LON,DIST+A,XO,YO,ZO)
C
C +-------------------------------------------------------------------+
C ! INNITIALIZE AREA AND NAVIGATION SPECIFIACTIONS.                   !
C +-------------------------------------------------------------------+
C
      CALL INIT(IA)
      INDX = 0
      SUMFLX = 0.0
      LINE = 1
      IEL = 1
      IF (CRTRCE) CALL TQMES('NL=$',NL)
      IF (CRTRCE) CALL TQMES('NEW=$',NEW)
      NSR = NSECL(NE)
C
C +-------------------------------------------------------------------+
C ! POSITION AT BEGINING OF LINE AND READ "NEW" NUMBER OF BRI-       !
C ! GTNESS VALUES INTO INBUF. THEN UNPACK IT AND PUT IT IN OUT-      !
C ! BUF.                                                             !
C +-------------------------------------------------------------------+
C
 356  ISEC = (LINE-1) * NSR
      CALL READA(IA,ISEC,NEW/3,INBUF)
      CALL CRACK(NEW,INBUF,IOUTBF)
C
C +-------------------------------------------------------------------+
C ! FOR EACH PIXEL IN THE LINE, COMPUTE EARTH AND VECTOR COORDI-     !
C ! NATES. NEXT COMPUTE THE AREA AND FLUX. AND FINALLY, INTEGRATE    !
C ! THE FLUX.                                                         !
C +-------------------------------------------------------------------+
C
  357 INDX = INDX + 1
      FSLIN = FLOAT(LINE*LINRES+ILIN-1)
      FSELE = FLOAT(IEL*IELRES+IELE-1)
      CALL SATEAR(PTIME,FSLIN,FSELE,FLAT,FLON,ITYPE,INAV
     *,BETAIN,BETDOT,ATFRAC)
```

```
      IF ((FLAT.EQ.100.0).AND.(FLON.EQ.200.0)) GOTO 300
      ICOUNT = IOUTBF(IEL)
C
      LAT = ILALO(FLAT)
      LON = ILALO(FLON)
      CALL CARTES(LAT,LON,A,XA,YA,ZA)
      AREA=GTAREA(FLAT,FLON)*(FLOAT(LINRES))*(FLOAT(IELRES))
      IF (AREA .EQ. 0.0) GOTO 300
C
      F = WEIGH(ICOUNT)
      SUMFLX = SUMFLX + FLUX(AREA,F)
  300 IEL = IEL + IEOFST
      IF (IEL.LE.NEW) GOTO 357
      IEL = 1
C
      IF((LINE/25)*25.EQ.LINE) CALL TQMES('PROCESSED LINE$',LINE)
      LINE = LINE + LNOFST
      IF (LINE.LE.NL) GOTO 356
 500  CONTINUE
C
      IF (TRACE) PRINT 198
 198  FORMAT(1X,'-->RESULT(S):')
      IF (TRACE) PRINT 199,SUMFLX
 199  FORMAT(' NET FLUX =',E14.7)
      ENCODE(72,99,NOUT) SUMFLX
 99   FORMAT('NET FLUX = ',E14.7)
      CALL TQ(NOUT)
      CALL TQMES('INDX=$',INDX)
C
      RETURN
      END


      FUNCTION WEIGH(ICOUNT)
C
C +----------------------------------------------------------------------+
C ! THIS FUNCTION COMPUTES THE WEIGHED RADIATIVE FLUX PER UNIT          !
C ! AREA WHEN GIVEN THE BRIGHTNESS VALUE FOR THE PIXEL.                  !
C +----------------------------------------------------------------------+
C
      TSUBG = 330.0 - (FLOAT(ICOUNT)/2.0)
      IF (ICOUNT.GT.176) TSUBG = FLOAT(418 - ICOUNT)
      WEIGH = 0.543*5.66E-08*TSUBG*TSUBG*TSUBG*TSUBG+44.54
      RETURN
      END


      SUBROUTINE INIT(IA)
C
C +----------------------------------------------------------------------+
```

```
C ! THIS ROUTINE SETS AREA AND NAVIGATION SPECIFICATIONS. IT CALLS
C ! COAREA, HOWBIG, GETNAV AND GETGAM.
C +--------------------------------------------------------------------------
C
      DIMENSION ISCRA(1200)
      LOGICAL TRACE
      COMMON /NAVNUM/ PTIME,BETAIN,BETDOT,ATFRAC,ITYPE,INAV
      COMMON /MAINIT/ NL,NE,ILIN,IELE,IDAY,JTIME,LINE,IEL
      COMMON /IRESOL/ LINRES,IELRES
      DATA TRACE /.TRUE./
      CALL COAREA(IA,IREEL,IDAY,JTIME,ILIN,IELE,LINRES,IELRES,IGE)
      CALL HOWBIG(IA,NL,NE)
C
      IF (TRACE) PRINT 92
 92   FORMAT(1X,'-->AREA SPECIFICATIONS:')
      IF (TRACE) PRINT 93,IA,IDAY,JTIME,ILIN,IELE,LINRES,IELRES,NL
 93   FORMAT(1X,'IA=',I1,' IDAY=',I7,' JTIME=',I6,' ILIN=',I4,
     *' IELE=',I4,' LINRES=',I3,' IELRES=',I3,' NL=',I3)
C
      PTIME = FTIME(JTIME)
      INAV = 1
      ITYPE = 1
      ATFRAC = 0.0
      CALL GETNAV(IDAY,IEXIST)
      CALL GETGAM(IDAY,JTIME,BETAIN,BETDOT)
      RETURN
      END


      SUBROUTINE CARTES(LAT,LON,DIST,X,Y,Z)
C
C +----------------------------------------------------------------------+
C ! THIS FUNCTION TRANSFORMS A POINT TO ITS VECTOR COMPONENTS.            !
C +----------------------------------------------------------------------+
C
      DATA DTR /0.0174532/
      XLAT = FLALO(LAT) * DTR
      XLON = FLALO(LON) * DTR
      X = DIST * COS(XLAT) * COS(XLON)
      Y = DIST * COS(XLAT) * SIN(XLON)
      Z = DIST * SIN(XLAT)
      RETURN
      END


      FUNCTION DOT(XL,YL,ZL,XM,YM,ZM)
C
C +----------------------------------------------------------------------+
C ! THIS FUNCTION TAKES THE SCALOR DOT PRODUCT OF THE TWO VECTORS.        !
C +----------------------------------------------------------------------+
```

```
C
      DOT = (XL*XM) + (YL*YM) + (ZL*ZM)
      RETURN
      END



      SUBROUTINE UNITVC(XM,YM,ZM,XU,YU,ZU)
C
C +---------------------------------------------------------------------+
C ! THIS SUBROUTINE COMPUTES THE UNIT VECTOR FOR THE GIVEN VECTOR       !
C +---------------------------------------------------------------------+
C
      COMMON /MAINIT/ NL,NE,ILIN,IELE,IDAY,JTIME,LINE,IEL
      DIVISR = SQRT((XM*XM) + (YM*YM) + (ZM*ZM))
      IF (DIVISR .EQ. 0.0) CALL TQMES('O DIVIDE..LINE =$',LINE)
      IF (DIVISR .EQ. 0.0) CALL TQMES('O DIVIDE..IEL =$',IEL)
      XU = XM/DIVISR
      YU = YM/DIVISR
      ZU = ZM/DIVISR
      RETURN
      END



      FUNCTION FLUX(AREA,F)
C
C +----------------------------------------------------------------------
C ! THIS FUNCTION COMPUTES THE FLUX FOR A PIXEL GIVEN ITS AREA AND
C ! RADIATIVE FLUX PER UNIT AREA.
C +----------------------------------------------------------------------
C
      LOGICAL TRACE
      COMMON /COMP/ XO,YO,ZO,XA,YA,ZA
      DATA TRACE /.FALSE./
      DATA INDX2 /0/
      INDX2 = INDX2 + 1
      XAO = XA - XO
      YAO = YA - YO
      ZAO = ZA - ZO
C
      IF (TRACE.AND.
     *(MOD(INDX2,250).EQ.0)) PRINT 222,XO,YO,ZO,XA,YA,ZA,XAO,YAO,ZAO
 222  FORMAT(9(1X,E14.7))
C
      CALL UNITVC(-XO,-YO,-ZO,XNO,YNO,ZNO)
C
      IF (TRACE.AND.
     *(MOD(INDX2,250).EQ.0)) PRINT 88,XNO,YNO,ZNO
 88   FORMAT(3(1X,E14.7))
C
      CALL UNITVC(XA,YA,ZA,XUA,YUA,ZUA)
```

```
C
      IF (TRACE.AND.
     *(MOD(INDX2,250).EQ.0)) PRINT 77,XUA,YUA,ZUA
 77   FORMAT(3(1X,E14.7))
C
      CALL UNITVC(XAO,YAO,ZAO,XUAO,YUAO,ZUAO)
C
      IF (TRACE.AND.
     *(MOD(INDX2,250).EQ.0)) PRINT 777, XUAO,YUAO,ZUAO
 777  FORMAT(3(1X,E14.7))
C
      XUOA = -XUAO
      YUOA = -YUAO
      ZUOA = -ZUAO
C
      IF (TRACE.AND.
     *(MOD(INDX2,250).EQ.0)) PRINT 66,XUOA,YUOA,ZUOA
 66   FORMAT(3(1X,E14.7))
C
      PROD = DOT(XUA,YUA,ZUA,XUOA,YUOA,ZUOA)
C
      IF (TRACE.AND.(MOD(INDX2,250).EQ.0)) PRINT 55,PROD
 55   FORMAT(1X,E14.7)
C
      PROD = PROD * DOT(XNO,YNO,ZNO,XUAO,YUAO,ZUAO)
C
      IF (TRACE.AND.
     *(MOD(INDX2,250).EQ.0)) PRINT 44, PROD
 44   FORMAT(1X,E14.7)
C
      PROD = ((PROD*AREA*F)/3.1419526)/(DOT(XAO,YAO,ZAO,XAO,YAO,ZAO))
C
      IF (TRACE.AND.
     *(MOD(INDX2,250).EQ.0)) PRINT 33, PROD,AREA,XAO,YAO,ZAO
 33   FORMAT(5(1X,E14.7))
C
      FLUX = PROD
      IF ((DOT(XNO,YNO,ZNO,XUOA,YUOA,ZUOA)).GT.0.0)
     *FLUX = 0.0
      IF ((DOT(XUA,YUA,ZUA,XUAO,YUAO,ZUAO)).GT.0.0)
     *FLUX = 0.0
      RETURN
      END


      FUNCTION ARETRI(A1,A2,A3,B1,B2,B3)
C
C
C +------------------------------------------------------------------
C ! THIS FUNCTION RETURNS THE AREA OF HALF THE PARALLELOGRAM ENCLOSED
C ! BETWEEN THE TWO GIVEN VECTORS.
```

```
C +-----------------------------------------------------------------------
C
      QI = (A2*B3) - (A3*B2)
      QJ = (A3*B1) - (A1*B3)
      QK = (A1*B2) - (A2*B1)
      ARETRI = (SQRT((QI*QI)+(QJ*QJ)+(QK*QK)))/2.0
      RETURN
      END


      FUNCTION GTAREA(XLAT1,XLON1)
C
C +-----------------------------------------------------------------+
C ! THIS FUNCTION IS MODIFIED SATSPO. THE AREA OF A PIXEL WHOSE     !
C ! TOP LEFT CORNER IS GIVEN, IS DETERMINED IN SQUARE METERS.       !
C +-----------------------------------------------------------------+
C
      COMMON /NAVNUM/ PTIME,BETAIN,BETDOT,ATFRAC,ITYPE,INAV
      DATA RADIUS /6372.10E03/
      CALL SATEAR(PTIME,XLIN,XELE,XLAT1,XLON1,2,INAV,BETAIN,BETDOT,0.
     *0)
      ILIN1=IROUND(XLIN)
      IELE1=IROUND(XELE)
      IF(ILIN1.EQ.0)GO TO 1
      ILIN2=ILIN1+1
      IELE2=IELE1
      ILIN3=ILIN1
      IELE3=IELE1+1
      ILIN4=ILIN2
      IELE4=IELE1+1
      XLIN1=ILIN1
      XELE1=IELE1
      XLIN2=ILIN2
      XELE2=IELE2
      XLIN3=ILIN3
      XELE3=IELE3
      XLIN4=ILIN4
      XELE4=IELE4
      CALL SATEAR(PTIME,XLIN1,XELE1,XLAT1,XLON1,1,INAV,BETAIN,BETDOT,0
     *0)
      CALL SATEAR(PTIME,XLIN2,XELE2,XLAT2,XLON2,1,INAV,BETAIN,BETDOT,0
     *0)
      CALL SATEAR(PTIME,XLIN3,XELE3,XLAT3,XLON3,1,INAV,BETAIN,BETDOT,0
     *0)
      CALL SATEAR(PTIME,XLIN4,XELE4,XLAT4,XLON4,1,INAV,BETAIN,BETDOT,0
     *0)
      IF(ABS(XLAT2).GT.90.0.OR.ABS(XLAT3).GT.90.0
     *.OR.ABS(XLAT4).GT.90.0) GOTO 1
      LAT1 = ILALO(XLAT1)
      LON1 = ILALO(XLON1)
      LAT2 = ILALO(XLAT2)
```

```
      LON2 = ILALO(XLON2)
      LAT3 = ILALO(XLAT3)
      LON3 = ILALO(XLON3)
      LAT4 = ILALO(XLAT4)
      LON4 = ILALO(XLON4)
      CALL CARTES(LAT1,LON1,RADIUS,RX1,RY1,RZ1)
      CALL CARTES(LAT2,LON2,RADIUS,RX2,RY2,RZ2)
      CALL CARTES(LAT3,LON3,RADIUS,RX3,RY3,RZ3)
      CALL CARTES(LAT4,LON4,RADIUS,RX4,RY4,RZ4)
      X13 = RX3 - RX1
      Y13 = RY3 - RY1
      Z13 = RZ3 - RZ1
      X12 = RX2 - RX1
      Y12 = RY2 - RY1
      Z12 = RZ2 - RZ1
      X42 = RX2 - RX4
      Y42 = RY2 - RY4
      Z42 = RZ2 - RZ4
      X43 = RX3 - RX4
      Y43 = RY3 - RY4
      Z43=RZ3-RZ4
      GTAREA = ARETRI(X12,Y12,Z12,X13,Y13,Z13)
     *+ ARETRI(X43,Y43,Z43,X42,Y42,Z42)
      RETURN
 1    GTAREA = 0
      RETURN
      END$
```

```
$FILEMA
DELETE GTFLUX,GORP
$CATALOG
NAME=GTFLUX,,R,W,D
TYPE=FG
LIB=NAVLIB,LL
BEGIN
$EOJ
```

Program Verification

Consider Ω, the solid angle which can be seen at an altitude h km above the earth as shown in Figure 7. This solid angle is formed by a cone of half angle ψ with apex at the observing point O.

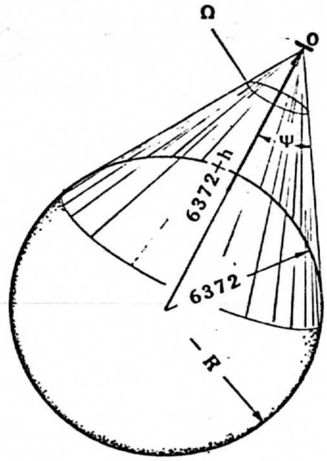


Figure 7. Visible solid angle

As suggested by a reviewer of a preliminary version of this paper, it is easy to show that the flux onto a flat plate of unit area at O is given by,

$$F = \int_0^{2\pi} \int_0^{\psi} \varepsilon B \sin\Theta \cos\Theta \, d\Theta \, d\phi \simeq \overline{\varepsilon B}\pi \sin^2\psi \tag{11}$$

A derivation is also given in the solved problems at the end of Chapter 5 in Fleagle and Businger's book cited above. The above expression, (11), assumes a uniform temperature over the earth. We have used this solution to check the computing program, including the vectors and area computations as well as the completeness of the radiance map. This was done by transforming the GOES data to a uniform count value of 120 using an enhancement program which leaves the data otherwise unchanged. This count value corresponds to 270K. Thus for the point at altitude 746km the value 166.58 watts m^{-2} would be expected. The value obtained was 167.08 Wm^{-2} (0.3% high). For the point at 161 km (11) gives 197.75 Wm^{-2} while we computed 196.67 (about 0.5% low).

Use of this test enabled us to find a coding error in the earlier version of this work which resulted in a substantial error. In the present version this error has been corrected by the insertion of line 361, what had been erroneously omitted. We are greatful to the reviewer who suggested this test.

It is concluded that the values obtained in this calculation are reasonable.

Results

Results in tabular form were already presented in the introduction. They are also shown in Figure 8. The first part of the flight is over relatively cloud-free ocean. Thus, the equivalent radiating surface has a relatively high temperature resulting in about 210 Wm^{-2} input. From t = 1350 onward this drops rapidly as the flight encounters high (cold) clouds to about 185 Wm^{-2}.

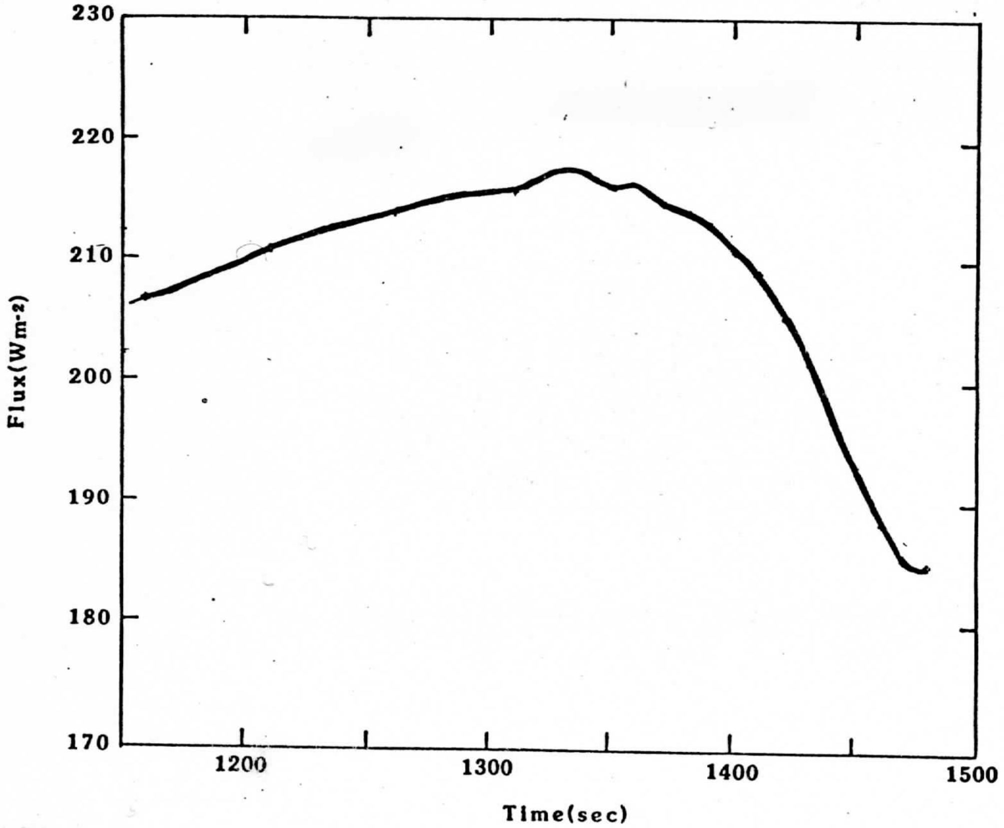


Figure 8. Flux as a function of time