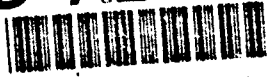


AD-A243 490



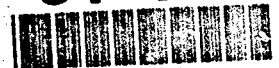
DTIC  
ELECTE  
DEC 17 1991  
S C D

A COMPUTER SOLUTION FOR MULTIPLE SATELLITE-SATELLITE AND  
SATELLITE-GROUND STATION VISIBILITY DETERMINATION

prepared by:

2Lt Jennifer L. Moore  
UTSI  
31 July 1991

91-17884

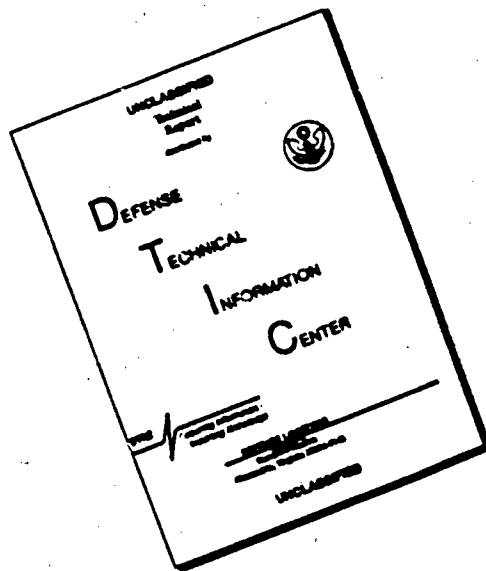


DISTRIBUTION STATEMENT A  
Approved for public release;  
Distribution Unlimited

91 1213 167

20000 901050

# DISCLAIMER NOTICE



**THIS DOCUMENT IS BEST  
QUALITY AVAILABLE. THE COPY  
FURNISHED TO DTIC CONTAINED  
A SIGNIFICANT NUMBER OF  
PAGES WHICH DO NOT  
REPRODUCE LEGIBLY.**

---

REPORT DOCUMENTATION PAGE

OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE 31 July 91	3. REPORT TYPE AND DATES COVERED THESIS/DOSSIER/BOOK	
4. TITLE AND SUBTITLE A Computer Solution for Multiple Satellite-Satellite and Satellite-Ground Station Visibility Determination			5. FUNDING NUMBERS	
6. AUTHOR(S) Jennifer L. Moore, 2d Lt				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AFIT Student Attending: University of Tennessee Space Institute			8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/CI/CIA-91-098	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFIT/CI Wright-Patterson AFB OH 45433-6583			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for Public Release IAW 190-1 Distributed Unlimited ERNEST A. HAYGOOD, Captain, USAF Executive Officer			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words)				
14. SUBJECT TERMS			15. NUMBER OF PAGES 50	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT	

ABSTRACT

The purpose of this study is to examine the problem of determining rise and set times for visibility periods of multiple satellite-satellite and satellite-ground station communications. Additionally, the ability to maintain constant communication links is explored.

This research examines three means to solve the problem: basic iteration, the Lawton Method, and an improved method involving parabolic blending and a closed form root solving technique. The greatest concentration is placed on the final solution and its application.

The result of this investigation is a computer program capable of determining rise and set times for as many as four satellites and four ground stations. In addition, the program performs a check for continuous communications between the indicated satellites and earth based resources.



Accession For

NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	

By \_\_\_\_\_

Distribution/

Availability Codes

Dist	Avail and/or	
A-1	Special	

## ACKNOWLEDGMENTS

This study represents the continued efforts of a project I began in a senior research study at the United States Air Force Academy. My task there was to research a computer method for rise/set determination between only two satellites using parabolic blending. My involvement with the project ended with the research, development, and early data collection. Two officers from the Academy, Lt Colonel Salvator Alfano and Captain Dave Negrón continued their own research into this topic after my graduation and have since submitted the results of our total effort for publication (See reference 1).

For this project, I resumed my work where my own research had ended a year before. The data published from Lt Colonel Alfano and Captain Negrón's continued efforts was very helpful in verifying the results of my own computer program. I am pleased to be able to bring to this project the added dimension of multiple satellite and ground station handling and continuous communications determination. I hope this paper provides insight into my own understanding of the problem and the course of my research as a whole. I am grateful to Lt Colonel Alfano and Captain Negrón for their continued support and encouragement in this endeavor.

## TABLE OF CONTENTS

CHAPTER	PAGE
I. INTRODUCTION . . . . .	1
II. VISIBILITY FUNCTION . . . . .	3
2.1 General Discussion . . . . .	3
2.2 Basic Rise/Set Determination . . . . .	4
III. THE LAWTON METHOD . . . . .	6
3.1 Fast Fourier Transform . . . . .	6
3.2 Regula-Falsi Method of Zero Finding . . . . .	7
IV. THE PARABOLIC BLENDING TECHNIQUE . . . . .	10
V. FINAL ALGORITHM . . . . .	12
5.1 Satellite Position Determination . . . . .	12
5.2 Ground Station Position Determination. . . . .	14
5.3 Rise/Set Time Determination. . . . .	16
5.4 Continuous Visibility Check. . . . .	18
VI. RESULTS . . . . .	21
VII. CONCLUSIONS . . . . .	24
LIST OF REFERENCES. . . . .	25
APPENDIX A: Program Run #1. . . . .	26
APPENDIX B: Program Run #2. . . . .	31
APPENDIX C: Program Run #3. . . . .	39
APPENDIX D: Program Listing . . . . .	49

LIST OF FIGURES

THE VISIBILITY FUNCTION . . . . .	4
REGULA FALSI METHOD . . . . .	8
PARABOLIC BLENDING. . . . .	10

## LIST OF SYMBOLS AND ABBREVIATIONS

a	Semi-major axis
$a_e$	Radius of the earth
$C_{max}$	Greatest modulus of the FFT
e	Eccentricity
$e_e$	Eccentricity of the earth
E	Eccentric anomaly
FFT	Fast Fourier Transform
i	Inclination
H	Altitude above mean sea level
$J_2$	Second harmonic of the earth's gravitational potential
JD	Julian Date
L	Geodetic Latitude
M	Mean anomaly
n	Mean motion
$n_0$	Mean motion at epoch
n	Anomalistic mean motion
p	Semi-latus rectum
P	Period of the visibility waveform
$R_E$	Radius of the earth
$r_1$	Position vector of satellite #1
$r_2$	Position vector of satellite #2
$T_u$	Centuries until 1 January 2000
UT	Universal Time
$\alpha_1$	Angle between earth's radius (perpendicular to tangent plane) and position vector #1, $r_1$



$\alpha_2$	Angle between earth's radius (perpendicular to tangent plane) and position vector $r_2$ .
$a_0, \dots, a_3$	Coefficients of the cubic equation defined by parabolic blending
$F$	Visibility function
$\delta$	East Longitude of the site
$\theta$	Local Sidereal Time
$\theta_g$	Greenwich Sidereal Time
$\theta_{g0}$	Greenwich Sidereal Time at 0 hours UT
$\phi$	Angle between $r_1$ and $r_2$
$v$	True anomaly
$\Omega$	Longitude of the ascending node
$w$	Argument of periapsis
$\omega_{max}$	Frequency of the largest modulus $C_{jmax}$

## CHAPTER 1

### INTRODUCTION

In space mission architecture, much rests on a satellite's ability to communicate with either an earth based tracking station or another satellite. Data relays as well as tracking, telemetry, and command directions depend on line of sight opportunities. Additionally, availability of communications determine satellite data rate requirements, data storage capabilities, and on orbit processing. All these factors affect satellite complexity, orbit selection and satellite constellation size. For these reasons, it is important to possess the ability to rapidly determine visibility rise and set times.

Three methods of determining the desired rise and set times are of interest to this study. The first propagates a basic visibility function then requires linear interpolation to find rise/set times. The second takes advantage of the fast Fourier transform to define characteristics of the visibility curve. This method makes use of the Regula Falsi iterative technique to determine rise/set times. The final method reconstructs the visibility curve using parabolic blending, then determines rise/set times using a closed form root solver. The process for determining visibility periods by this third method is detailed. The final result is a computer program capable of handling multiple satellites and ground stations, determining whether or not continuous

communications among the satellites and with the ground is possible, and completing the task in a timely manner.

## CHAPTER 2

### VISIBILITY FUNCTION

#### 2.1 General Discussion

The concept of the visibility function is central to any method of determining rise/set times for satellite-satellite or satellite-ground station in-view periods. The visibility function involves a line of sight calculation between the two objects of interest. The value of the function varies with time in accordance to the changing positions of the two objects.

For two satellites in orbit around the earth, line of sight is possible only when both objects are above a plane tangent to the earth's surface. The most extreme positions for the satellites which still allow for line of sight occur when both lie in the tangent plane. These positions are defined by the following equations<sup>4</sup>:

$$a_1 = \cos^{-1}(R_1/r_1)$$

$$a_2 = \cos^{-1}(R_1/r_2)$$

The angle between the two position vectors at any point is defined as:

$$\phi = \frac{(r_1 \cdot r_2)}{|r_1||r_2|}$$

The visibility function is then defined by<sup>5</sup>:

$$\Gamma = a_1 + a_2 - \phi$$

When the value of the visibility function is positive, the satellites have line of sight. Similarly, a negative value indicates no line of sight. A value of zero is either a

rise or set time for visibility (See Figure 1). The same visibility function applies to a satellite and a ground station.

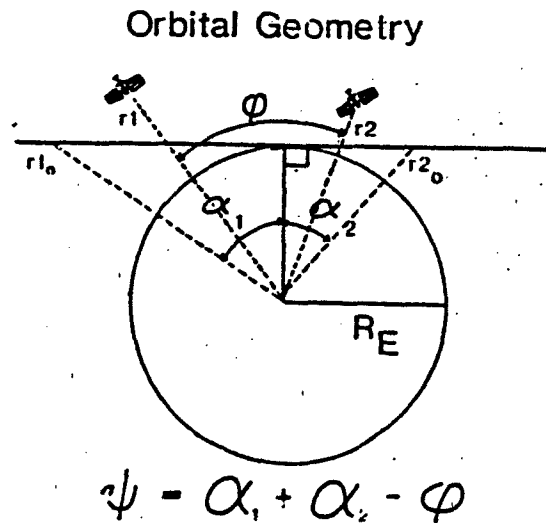


Figure 1

## 2.2 Basic Rise/Set Determination

The most direct method of determining visibility opportunities is small time step progression of the visibility function. At each step, the value of the visibility function is calculated. When plotted versus time, the visibility function yields a visibility waveform. When a sign change occurs in the value of the function and the wave crosses the x axis, a rise or set time is indicated. Linear interpolation is used to determine the exact time of the crossing. Although this method is legitimate, it is cumbersome and time consuming. The small time step necessary to insure the best solutions using

linear interpolation leads to large data sets and an extremely inefficient number of calculations by the computer. In order to determine the rise/set time for two satellites over a period of twenty-four hours with a time step of five seconds, more than seventeen-thousand calculations of the visibility function alone must be computed. As a method for real time solutions, this process is simply not realistic.

## CHAPTER 3

### THE LAWTON METHOD

#### 3.1 The Fast Fourier Transform

In his paper, J. A. Lawton presents a method for determining visibility opportunities using a fast Fourier transform, FFT, to estimate the period of the visibility waveform<sup>5</sup>. The use of the FFT is in response to the need to process large quantities of data points with fewer calculations while still maintaining the accuracy of the interpolating trigonometric polynomial<sup>4</sup>. In his presentation, Lawton uses the FFT to select the most prominent modulus of the waveform from the equation<sup>5</sup>

$$C_j = \sum_{k=1}^n \Gamma(t_k) e^{-i(2\pi j k/n)} \quad j = 1, 2, \dots, n$$

where

$\Gamma(t_k)$  = the visibility function at time  $t_k$

$i = \sqrt{-1}$

$n$  = number of data points

The value of  $C_j$  with the largest modulus,  $C_{jmax}$ , is then used to determine the value of the period of the waveform from the following equations<sup>5</sup>.

$$w_{jmax} = \frac{2\pi(j_{max} - 1)}{n}$$

$$P = \frac{2\pi}{w_{jmax}}$$

Once the period of the waveform is determined, it is used in conjunction with a curve searching technique to determine the rise/set times of the objects in question.

### 3.2 The Regula-Falsi Method of Zero Finding

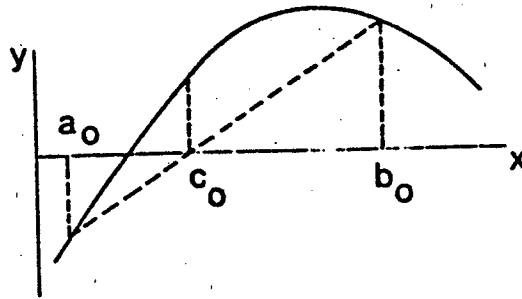
The visibility waveform is first sectioned by fractions of the period determined by the FFT. The slope of the visibility function is determined in each progressive section of the wave until a local maximum value of the function is isolated between a positive and negative slope value. If the local maximum value is found to be positive, indicating line of sight is possible, then the portion of the curve in that particular section is searched for rise and set times. If the value of the visibility function is negative at the local maximum, the searching process resumes to isolate the next local maximum value.

Curve searching in this process is accomplished using the Regula Falsi method of zero finding. The equations are:

$$c_0 = \frac{a_0 f(b_0) - b_0 f(a_0)}{f(b_0) - f(a_0)}$$



An illustration of the process is shown below in Figure 2.



$$c_0 = \frac{a_0 f(b_0) - b_0 f(a_0)}{f(b_0) - f(a_0)}$$

Figure 2

If the value of  $f(c_0)$  is zero, the search is complete. Otherwise, an iterative technique begins with reassignment of the variables occurring as follows<sup>7</sup>:

if  $f(a_0)f(c_0) < 0$ , then  $a_1 = a_0$  and  $b_1 = c_0$

if  $f(a_0)f(c_0) > 0$ , then  $a_1 = c_0$  and  $b_1 = b_0$

The above reassignment continues and the value of  $c$  is recalculated until  $f(c_0)$  is equal to zero. In the initial process of the curve examination, this method is used to determine the time associated with a local maximum in order to determine if the value of the visibility function at the local maximum is positive or negative. In this case, no crossing of the  $x$  axis takes place and the iteration of the Regula Falsi method continues until  $a_0 = b_0$  or until the difference between  $a_0$  and  $b_0$  is sufficiently small.

Although the Lawton process is sound in the determination of rise/set times, it has the disadvantage of being limited to orbits with eccentricities less than  $0.10^5$ . This is necessary to ensure a relatively periodic waveform. Very low eccentricity or nearly circular orbits produce nearly sinusoidal waves while highly elliptical orbits produce wave forms much more difficult to predict. Evidence of this can be found in the figures included with the results to this work. Aperiodic waveforms may result in inaccurate determination of the period and make curve searching a more difficult task. Additionally, this method is only applicable to evenly spaced data points<sup>4</sup>. Sparse data or unevenly spaced points cannot be used with the fast Fourier transform.

In his work, Lawton states this method accounted for 99.997% of the rise/set times determined by the iteration method described previously using a spherical-Earth gravity field and 99.992% of the rise/set times in an oblate-Earth gravity field.<sup>5</sup> He fails to state the accuracy of the rise/set times found by this method as compared to the basic iteration technique. The work in this study did not include reproduction of his results nor investigation of the accuracy of this method.

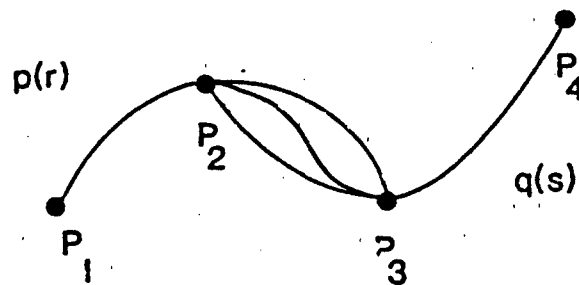
## CHAPTER 4

### THE PARABOLIC BLENDING TECHNIQUE

Parabolic blending is a method of curve fitting which maintains continuity of two parabolas formed from four data points. The first parabola,  $p(r)$  is defined by points 1, 2 and 3, while the second,  $q(s)$  includes points 2, 3, and 4. The parabolically blended curve is formed between points 2 and 3 (See Figure 3) and is defined by<sup>9</sup>

$$C(t) = (1 - t)p(r) + tq(s)$$

$$\text{where } 0 \leq t \leq 1$$



$$C(t) = (1 - t)p(r) + tq(s)$$

Figure 3

At  $t = 0$ ,  $C(t)$  has the same slope as  $p(r)$  while at  $t = 1$ ,  $q(s)$  is the influential curve. The final result of the blending is a cubic maintaining characteristics of both

parabolas. The equation of the curve is written as<sup>1,9</sup>

$$C(t) = a_3t^3 + a_2t^2 + a_1t + a_0$$

where  $a_0 = P_2$

$$a_1 = -.5P_1 + .5P_3$$

$$a_2 = P_1 - 2.5P_2 + 2P_3 - .5P_4$$

$$a_3 = -.5P_1 + 1.5P_2 - 1.5P_3 + .5P_4$$

This technique differs from the cubic spline since the equation of the blended curve is independent of all other data points except for the four points used to define the two parabolas<sup>9</sup>. Also, parabolic blending works well with both equally and unequally space data points<sup>1</sup>.

## CHAPTER 5

### FINAL ALGORITHM

The final result of this study is a computer program designed to give the user the option of determining the visibility opportunities between a combination of up to four satellites and four ground stations. More than four satellites or ground stations requires significantly more calculations and data storage. The step-by-step process of rise/set determination is detailed in the sections which follow.

#### 5.1 Satellite Position Determination

To begin the process of determining the rise/set times for satellite-satellite or satellite-ground station visibility, it is necessary to calculate the position vector of the satellite at the start of the time of interest. This requires that six critical orbital elements be known for this calculation. They are the mean motion ( $n$ ), inclination ( $i$ ), eccentricity ( $e$ ), longitude of the ascending node ( $\Omega$ ), argument of periapsis ( $w$ ), and the mean anomaly ( $M$ ). Although the position vector can be determined from a variety of elements, these six were chosen since they are readily known from an ephemeris of satellite information. These elements and the position of the satellite are determined with respect to the geocentric reference frame.

The first step in the determination of the position vector is the calculation of the eccentric anomaly. This is

completed through the use of the Newton-Rhapson Iteration.

The equations are<sup>2</sup>:

$$M_n = E_n - e \text{ SIN}(E_n)$$

$$E_{n+1} = E_n + \frac{M - M_n}{1 - e \text{ COS}(E_n)}$$

For the first iteration, M is used in the place of  $E_n$ . When  $E_{n+1}$  is calculated, a comparison is made between  $E_n$  and  $E_{n+1}$ . When the difference is sufficiently small, the iteration is stopped. For the final computer program, the tolerance was set for a difference less than 0.000001. Convergence of the iteration may be of concern for orbits with eccentricities approaching 1.0<sup>2</sup>. Once determined, the final value of  $E_n$  is used in the calculation of the true anomaly as follows<sup>2</sup>:

$$\text{COS}(v) = \frac{e - \text{COS}(E)}{e \text{ COS}(E) - 1}$$

$$\text{SIN}(v) = \frac{a(1 - e^2)}{p} \cdot \text{SIN}(E)$$

$$p = a(1 - e \text{ COS}(E))$$

Once the proper quadrant of  $v$  is determined, the position vector is calculated in the PQW or perifocal reference frame<sup>2</sup>.

$$r = r \text{ COS}(v) P + r \text{ SIN}(v) Q$$

$$r = \frac{a(1 - e^2)}{1 + e \text{ COS}(v)}$$

A coordinate transformation is then accomplished to place the position vector in the geocentric frame<sup>3,10</sup>.

The above process determines the position vector of the satellite at a given time. Once this is accomplished, the position of the satellite must be propagated through the orbit by some time step dictated by the user. This is accomplished through the use of the following equations<sup>1</sup>:

$$n = n_0 \left[ 1 + \frac{1.5J_2(1-e^2)}{p^2} (1 - 1.5\text{SIN}^2(i)) \right]$$

$$\Omega = -\left( \frac{1.5J_2 \text{COS}(i)}{p^2} \right) n$$

$$w = \left( \frac{1.5J_2 [2 - 2.5\text{SIN}^2(i)]}{p^2} \right) n$$

These changes in the values of the longitude of the ascending node and the argument of periapsis represent secular perturbations resulting from an aspherical earth.<sup>6</sup> Once these elements are propagated forward, the position vector is recalculated. In order to decrease compounding error, the position vector is propagated each iteration from the original values of  $\Omega$  and  $w$ .

It is important to note other perturbing forces including atmospheric drag were not taken into account for this study. Had drag been included, secular changes in the semi-major axis, eccentricity and the inclination would have been taken into account.<sup>6</sup>

## 5.2 Ground Station Position Determination

For the determination of satellite-ground station visibility periods, the position of the ground station must be calculated. This requires the user know the following information: the year, month, day, hour, minute, and second

the simulation is to begin, the east longitude and geodetic latitude of the site, and the altitude of the site above mean sea level. This information is used in the determination of the Julian Date, the local sidereal time of the site, and the final calculation of the position vector.

This process begins with the determination of the Julian Date given the year, month, date, and time. The Julian Date is referenced to noon, 1 January 4713 B.C. The equation for the Julian Date used in this program only applies to dates between 1 March 1900 and 28 February 2100<sup>3,10</sup>. The equation is<sup>3</sup>:

$$JD = 367 * Year - \text{int}\left(\frac{7 * (Year + \text{int}(\frac{Month+9}{12}))}{4}\right) + \text{int}\left(\frac{275 * Month}{9}\right) + Day + 1721013.5 + \frac{\text{int}(\frac{Second + Min}{60})}{24} + Hour$$

The Julian Date is then used to calculate the Greenwich Sidereal Time at 0 hours UT<sup>3</sup>:

$$\theta_{g0} = 100.4606184 + 36000.77004T_u + .00038793T_u^2$$

$$\text{where } T_u = \frac{\text{Julian Date} - 2451545.0}{36525.0}$$

The Greenwich Sidereal Time is then calculated according to the rotation of the earth and the universal time in terms of angles by<sup>2</sup>:

$$\theta_g = \theta_{g0} + \omega_e(t - t_0)$$



Finally, the Local Sidereal Time for the site is determined by in terms of angles<sup>2</sup>:

$$\theta = \theta_g + \delta t$$

From this point, the position vector of the site in the geocentric frame is calculated from<sup>2</sup>:

$$X = \left[ \frac{a_e}{\sqrt{1 - e_e^2 \sin^2(L)}} + H \right] \cos(L)$$

$$Z = \left[ \frac{a_e (1 - e_e^2)}{\sqrt{1 - e_e^2 \sin^2(L)}} + H \right] \sin(L)$$

$$\vec{r} = X \cos(\theta) \mathbf{I} + X \sin(\theta) \mathbf{J} + Z \mathbf{K}$$

In a manner similar to the satellite position vector propagation, the site position vector is propagated ahead with respect to the rotation of the earth and the time step indicated. It should be noted that both the satellite position vector and the ground site position vector may be calculated using either a spherical earth or an oblate earth in accordance to the users choice. If an oblate earth is chosen, the K component of both position vectors is scaled by a factor of  $1/(1 - e_e^2)^{1.5}$ .

### 5.3 Rise/Set Time Determination

Once the position vectors of the two objects of interest have been determined, the visibility function is calculated at each time step. Although any time step may be specified by the program user, it is suggested 25 second steps be used to allow for computer efficiency. Extremely

large time steps may result in missed rise/set times. In order to insure complete evaluation of the entire time of interest, the position vectors and visibility function are calculated for times one step earlier and one step later than specified by the user. Starting at time equal to zero, parabolic blending is applied to the first four visibility function data points. The equation of the curve between points 2 and 3 is calculated. The real roots of the resulting cubic equation are computed using a closed form root solver<sup>6</sup>. The equations to compute the real roots of a quadratic are included in the program in the event the blended curve is a second degree polynomial or a straight line. All complex roots are ignored for this process. As stated previously, the blended curve is defined by<sup>5</sup>:

$$C(t) = (1 - t)P(r) + tq(s)$$

where  $t$  may take any value between 0 and 1. For this reason, only roots between and including 0 and 1 are considered.

When a real root meeting the above requirements is determined, the coefficients  $a_0$ ,  $a_1$ ,  $a_2$ , and  $a_3$  are recalculated using the times corresponding to points 1, 2, 3 and 4. The roots of the original blended equation are then substituted into this second blended equation. The resulting value of the blended equation is a rise or set time. The root is then increased by a small increment and the value of the second equation is recalculated. If the value is positive, then the root in question represents a

rise time. Similarly, a negative value represents a set time.

The next blended curve is calculated between points 3 and 4 and then checked for roots. This process continues until the entire visibility waveform has been evaluated. The result is a rise/set history for two objects over a given time period.

#### 5.4 Continuous Visibility Check

As previously mentioned, the final computer program offers the user the option of selecting up to four satellites and four ground stations for evaluation of rise/set times. It is the purpose of the continuous visibility check to evaluate the rise/set history in order to determine if constant communication is possible among the given satellites or between the ground and a constellation of satellites.

There are basically two types of communication links for satellite communications. They are forward/reverse links which occur between satellites, and uplink/downlink occurring between a satellite and the ground<sup>11</sup>. For two satellites, forward/reverse continuous communication is not possible if a rise or set occurs or if the satellites were not in constant communication from the beginning of the evaluation. For a three satellite constellation, at least two paths of visibility out of a possible three must be maintained at all times for constant communication to be

possible. Similarly, four satellites require three paths of visibility out of a possible six. Additionally, checks are made to determine if each ground station is in view of at least one satellite at all times for the purpose of uplink/downlink capabilities.

The continuous visibility check is accomplished by constructing a table containing every rise and set time for each possible communications path. For example, in a three satellite constellation, rise/set times are determined between satellites #1 and #2, #1 and #3, and #2 and #3. The table for this constellation would include all the rise and set times from these satellites. These rise/set opportunities represent critical times since they are the only times during the simulation when the status of visibility can change. In order to accomplish the continuous visibility check, the status of each visibility path must be determined at every rise set time. For the program, a (1) was used if two satellites were visible or experiencing a rise at a given time. Similarly, a (0) was used for a state of no visibility or a set. An example of a three satellite table is shown below.

Time (sec)	Sat's #1 & #2	Sat's #1 & #3	Sat's #2 & #3
0	1	0	1
128	1	0	0
356	0	0	1
700	0	1	0
825	0	1	0
967	1	1	0

As mentioned before, for constant communication to be maintained for a three satellite constellation, at least two paths of visibility must be available at all times. That means the sum of the (1)'s and (0)'s at a given time in the table must be greater than or equal to two. In the example above, communication is broken at 128 seconds and is not re-established until 967 seconds.

The same process is used for a four satellite constellation. In such a case, there are six possible communication paths, and the sum at any given rise or set time must be greater than or equal to three. For ground station to satellite communication, the ground station must be able to communicate with at least one satellite at any time. Therefore, the sum at each rise and set time must be greater than or equal to one for constant communication to be maintained.

## CHAPTER 6

### RESULTS

Attached are the results of three different runs of the final computer program attached in Appendix D. Run #1 represents a validation of the rise/set algorithm of the program. The results of this program were compared to the results contained in the research published in reference one. The initial orbital elements of the four satellites used in the program are found preceding the rise/set times. This run was achieved over a time of 86,400 seconds and a 250 second time step with a spherical earth. These results were almost identical to those achieved by the separate program created by Alfano and Negrón<sup>1</sup>. Also included with the first run results are the rise/set values calculated using the basic iteration method mentioned earlier in the research. For this basic method, a 10,000 second time span was evaluated in 5 second intervals. The greatest variance in the compared results was less than 1 second. It should be noted the computer time to generate the visibility curve for the 10,000 second basic method between two satellites was 12 minutes, while only 3 minutes were required using the parabolic blending method over a 86,100 seconds with a 250 second time step. Run #1 also included a results using a ground station and one satellite. This run involved a 125 second time step over 86,400 seconds and an oblate earth. These results are also found in Appendix A and compare almost identically to those achieved by Alfano and Negrón<sup>1</sup>.

The basic iteration method was applied to this data as well and the results can be found in the adjoining column. The greatest variation between the parabolic blending method and the basic iteration method in this case was approximately 3 seconds.

The results of the second run are included in Appendix B. These results use the same satellites and the ground station from run #1 but all calculations involved the use of an oblate earth. As expected, the oblate earth resulted in earlier rise times and later set times<sup>1</sup>. Additionally, the continuous visibility check was employed in this run. Plots of the visibility function of the ground station versus each of the four satellites is included in this Appendix. Additionally, the combined plot of all four satellite-ground station functions are included as well as the six possible visibility combinations of the four satellites. These combination plots can be used to verify the results of the continuous visibility check.

Appendix C contains data resulting from the third run of the program. This run was intended to check the complete capabilities of the program by using four different satellites and four different ground stations. The run was accomplished over a time span of 86,400 seconds at a time step of 125 seconds. Data for the satellites was chosen randomly from an orbital element ephemeris of actual satellites provided by the United States Space Command. The four ground stations chosen are Space Shuttle tracking

sites. They include<sup>s</sup>: (=1) Kaena Point, Hawaii; (=2) Mahe Island, Indian Ocean; (=3) Thule, Greenland; (=4) Vandenburg Air Force Base, California. Again, the plots of the visibility functions are included and may be used to verify the results of the visibility check. The total run time for this case was approximately twenty minutes.



## CHAPTER 7

### CONCLUSIONS

While the basic iterative technique and Lawton's method are viable solutions to the rise and set time problem, the use of parabolic blending and a closed form solution has many advantages. First, parabolic blending is a flexible means to reconstruct the visibility curve. It can be used with sparse or unequally spaced data and is not restricted to near circular orbits. Second, a closed form root solver allows rise/set times to be calculated directly rather than by an iterative technique thereby reducing computer run time. The resulting timely determination of rise/set times makes this method a useful tool for use with multiple satellites and ground stations. Additionally, the inclusion of a continuous visibility check makes this program a useful tool for the investigation of satellite requirements and space mission design.

## LIST OF REFERENCES

1. Alfano, S., D. Negron, Jr., and Jennifer L. Moore. "Rapid Determination of Satellite Visibility Periods." (being reviewed for publication)
2. Bate, R. R., D. D. Mueller, and J. E. White. Fundamentals of Astrodynamics. New York: Dover, 1971.
3. Buglia, James J. "Compilation of Methods in Orbital Mechanics and Solar Geometry," NASA Reference Publication 1204, 1988.
4. Burden, Richard L. and J. Douglas Faires. Numerical Analysis. Boston: PWS-KENT, 1989.
5. Cook, James B. Orbital Trajectory Data for STS Standard Missions. Houston: McDonnell Douglas Technical Services Co, 1978.
6. Escobal, Pedro R. Methods of Orbit Determination. Malabar: Kreiger, 1965.
7. Kreyszig, Erwin. Advanced Engineering Mathematics. New York: John Wiley and Sons, 1965.
8. Lawton, J. A. "Numerical Method for Rapidly Determining Satellite-Satellite and Satellite-Ground Station In-View Periods," (U.S. Naval Weapons Center, Dahlgren VA, USA) J Guid Control Dyn v10 n1 Jan-Feb 1987, 32-36.
9. Roger, David F., and J. Alan Adams. Mathematical Elements for Computer Graphics. New York: McGraw-Hill,
10. Vallado, David. Methods of Astrodynamics: A Computer Approach. United States Air Force Academy, 1990.
11. Wertz, James R. and Wiley J. Larson. Space Mission Analysis and Design. Boston: Dordrecht, 1989.

Appendix A: Program Run #1

Longitude of the Ascending Node : 0.00000000 DEG  
Argument of Periapsis : 0.00000000 DEG

INITIAL ORBITAL ELEMENTS OF SATELLITE #2:

Eccentricity : 0.9363060000  
Inclination : 64.98740000 DEG  
Mean Anomaly : 0.00000000 DEG  
Longitude of the Ascending Node : 0.00000000 DEG  
Argument of Periapsis : 0.00000000 DEG

INITIAL ORBITAL ELEMENTS OF SATELLITE #3:

Eccentricity : 0.0078742000  
Inclination : 82.87090000 DEG  
Mean Anomaly : 0.00000000 DEG  
Longitude of the Ascending Node : 0.00000000 DEG  
Argument of Periapsis : 0.00000000 DEG

INITIAL ORBITAL ELEMENTS OF SATELLITE #4:

Eccentricity : 0.0048964000  
Inclination : 144.64140000 DEG  
Mean Anomaly : 0.00000000 DEG  
Longitude of the Ascending Node : 0.00000000 DEG  
Argument of Periapsis : 0.00000000 DEG

Rise/Set Times between Satellite #1 and Satellite #2:  
83679.13091 SEC - SET

Rise/Set Times between Satellite #1 and Satellite #3:

1450.33207 SEC - SET	1450.222816 SEC
3965.41739 SEC - RISE	3965.454545 SEC
6889.49771 SEC - SET	6889.599589 SEC
9371.13568 SEC - RISE	9371.062356 SEC
12399.31335 SEC - SET	
14774.02357 SEC - RISE	
18419.90807 SEC - SET	
20218.05526 SEC - RISE	
25546.76967 SEC - SET	
27789.37858 SEC - RISE	
30949.66914 SEC - SET	
33360.02044 SEC - RISE	
36352.15032 SEC - SET	
39814.44584 SEC - RISE	
41760.11003 SEC - SET	
44235.69347 SEC - RISE	
47181.56457 SEC - SET	
49643.34461 SEC - RISE	
52636.48892 SEC - SET	
55045.12369 SEC - RISE	
58209.92219 SEC - SET	
60445.55653 SEC - RISE	
65807.11706 SEC - SET	
67619.78663 SEC - RISE	
71227.16594 SEC - SET	
73604.36321 SEC - RISE	
76628.56161 SEC - SET	
79111.00284 SEC - RISE	
82033.93906 SEC - SET	
84549.30712 SEC - RISE	

7692.53382 SEC - SET  
 9900.74198 SEC - RISE  
 13747.34515 SEC - SET  
 13662.27191 SEC - RISE  
 19034.87777 SEC - SET  
 21382.21557 SEC - RISE  
 25088.48497 SEC - SET  
 27134.52567 SEC - RISE  
 30810.51947 SEC - SET  
 32961.40487 SEC - RISE  
 36817.71133 SEC - SET  
 38842.89352 SEC - RISE  
 42484.01314 SEC - SET  
 44737.10033 SEC - RISE  
 48377.75627 SEC - SET  
 50604.70112 SEC - RISE  
 54259.69364 SEC - SET  
 56413.52413 SEC - RISE  
 60087.93877 SEC - SET  
 62158.24402 SEC - RISE  
 65841.04035 SEC - SET  
 67888.41159 SEC - RISE  
 71560.14573 SEC - SET  
 73673.39260 SEC - RISE  
 77319.73512 SEC - SET  
 79526.35809 SEC - RISE  
 83147.81813 SEC - SET  
 85412.55717 SEC - RISE

7692.62000 SEC

Rise/Set Times between Satellite #2 and Satellite #3:

2997.87843 SEC - SET	2997.887324 SEC
5931.10222 SEC - RISE	5931.141593 SEC
8939.96295 SEC - SET	8939.946429 SEC
11435.44659 SEC - RISE	
14472.78852 SEC - SET	
16877.79953 SEC - RISE	
19934.36110 SEC - SET	
22297.74182 SEC - RISE	
25367.33066 SEC - SET	
27706.27682 SEC - RISE	
30785.29945 SEC - SET	
33108.06577 SEC - RISE	
36194.14202 SEC - SET	
38505.34408 SEC - RISE	
41597.01520 SEC - SET	
43899.47815 SEC - RISE	
46995.62990 SEC - SET	
49291.29522 SEC - RISE	
52391.13775 SEC - SET	
54681.30707 SEC - RISE	
57784.28440 SEC - SET	
60069.96329 SEC - RISE	
63175.51700 SEC - SET	
65457.45070 SEC - RISE	
68565.29108 SEC - SET	
70844.04327 SEC - RISE	
73953.83055 SEC - SET	
76229.83982 SEC - RISE	
79341.37564 SEC - SET	
81615.05419 SEC - RISE	
84728.03935 SEC - SET	

13412.02208 SEC - RISE  
 17526.54531 SEC - SET  
 19673.27556 SEC - RISE  
 23787.20492 SEC - SET  
 25923.63498 SEC - RISE  
 30032.62420 SEC - SET  
 32169.13711 SEC - RISE  
 36275.30607 SEC - SET  
 38411.96192 SEC - RISE  
 42516.36090 SEC - SET  
 44653.14374 SEC - RISE  
 48756.32667 SEC - SET  
 50893.22948 SEC - RISE  
 54995.50864 SEC - SET  
 57132.55012 SEC - RISE  
 61234.12410 SEC - SET  
 63371.29703 SEC - RISE  
 67472.30883 SEC - SET  
 69609.58623 SEC - RISE  
 73710.14286 SEC - SET  
 75847.56839 SEC - RISE  
 79947.69212 SEC - SET  
 82085.26989 SEC - RISE  
 86185.16424 SEC - SET

Rise/Set Times between Satellite #3 and Satellite #4:

728.00260 SEC - SET	728.253012 SEC
2144.97084 SEC - RISE	2145.755968 SEC
3508.89011 SEC - SET	3508.638444 SEC
34217.02408 SEC - RISE	
35246.56198 SEC - SET	
36889.15759 SEC - RISE	
38274.52867 SEC - SET	
39624.21934 SEC - RISE	
41174.78140 SEC - SET	
42839.99372 SEC - RISE	
43670.74866 SEC - SET	
72091.70409 SEC - RISE	
72474.06899 SEC - SET	
74570.33847 SEC - RISE	
75735.76905 SEC - SET	
77179.52589 SEC - RISE	
78783.75035 SEC - SET	
80215.31302 SEC - RISE	
81418.26446 SEC - SET	
83425.21845 SEC - RISE	
83937.59649 SEC - SET	

THE LOCATION OF GROUND STATION =1  
 EAST LONGITUDE: 0.000000  
 GEODETIC LATITUDE: 39.000000  
 ALTITUDE ABOVE THE EARTH ELLIPSOID: 2.900000

INITIAL ORBITAL ELEMENTS OF SATELLITE =1:  
 Eccentricity : 0.0078742000  
 Inclination : 82.87090000 DEG  
 Mean Anomaly : 0.00000000 DEG  
 Longitude of the Ascending Node : 0.00000000 DEG  
 Argument of Periapsis : 0.00000000 DEG

RISE AND SET TIMES BETWEEN GROUND STATION =1 AND		SATELLITE =1:
366.33816 SEC - RISE		366.4128596 SEC
793.74516 SEC - SET		793.3902878 SEC
5973.86027 SEC - RISE		5972.4175820 SEC
6041.56945 SEC - SET		6041.6470590 SEC
39460.44479 SEC - RISE		
39949.95608 SEC - SET		
44907.37075 SEC - RISE		
45292.93512 SEC - SET		
86261.28564 SEC - RISE		

Appendix B: Program Run #2



Rise/Set Times between Satellite #1 and Satellite #3:

1460.52842 SEC - SET  
3954.59411 SEC - RISE  
6901.04371 SEC - SET  
9357.44428 SEC - RISE  
12414.55102 SEC - SET  
14750.81501 SEC - RISE  
18432.45193 SEC - SET  
20059.68746 SEC - RISE  
25578.36179 SEC - SET  
27770.73291 SEC - RISE

Rise/Set Times between Satellite #1 and Satellite #4:

1807.91575 SEC - SET  
4069.37257 SEC - RISE  
7695.00886 SEC - SET  
9900.04560 SEC - RISE  
13549.39377 SEC - SET  
15662.26623 SEC - RISE  
19335.91349 SEC - SET  
21381.74668 SEC - RISE  
25066.47341 SEC - SET  
27132.76382 SEC - RISE

Rise/Set Times between Satellite #2 and Satellite #3:

2997.97685 SEC - SET  
5924.02061 SEC - RISE  
8944.66759 SEC - SET  
11427.18027 SEC - RISE  
14479.19061 SEC - SET  
16868.98772 SEC - RISE  
19941.62420 SEC - SET  
22298.63982 SEC - RISE  
25375.13704 SEC - SET  
27696.99684 SEC - RISE

Rise/Set Times between Satellite #2 and Satellite #4:

4932.42624 SEC - SET  
7104.92257 SEC - RISE  
11273.16055 SEC - SET  
13407.13572 SEC - RISE  
17537.31159 SEC - SET  
19669.08990 SEC - RISE  
23788.08220 SEC - SET  
25919.79045 SEC - RISE

Rise/Set Times between Satellite #3 and Satellite #4:

738.80937 SEC - SET  
2137.92179 SEC - RISE  
3816.84185 SEC - SET

\*

(NOTE: NO ENTRIES OR MESSAGE BELOW INDICATES CONTINUOUS COMMUNICATIONS.)

Communications broken at 3516.84185 SEC  
Communications re-established at 3954.59411 SEC  
Communications broken at 8944.66759 SEC  
Communications re-established at 9357.44428 SEC  
Communications broken at 14479.19061 SEC  
Communications re-established at 14750.81501 SEC  
Communications broken at 19335.91349 SEC  
Communications re-established at 19669.08990 SEC  
Communications broken at 19941.62420 SEC  
Communications re-established at 20069.68746 SEC  
Communications broken at 25375.13704 SEC  
Communications re-established at 27132.76382 SEC

RISE AND SET TIMES BETWEEN GROUND STATION #1 AND

SATELLITE #1:

35

RISE AND SET TIMES BETWEEN GROUND STATION #1 AND

SATELLITE #2:

200.76093 SEC - RISE

3060.09662 SEC - SET

13147.673 SEC - RISE

RISE AND SET TIMES BETWEEN GROUND STATION #1 AND

SATELLITE #3:

366.33816 SEC - RISE

793.74516 SEC - SET

5973.86027 SEC - RISE

6041.56945 SEC - SET

RISE AND SET TIMES BETWEEN GROUND STATION #1 AND

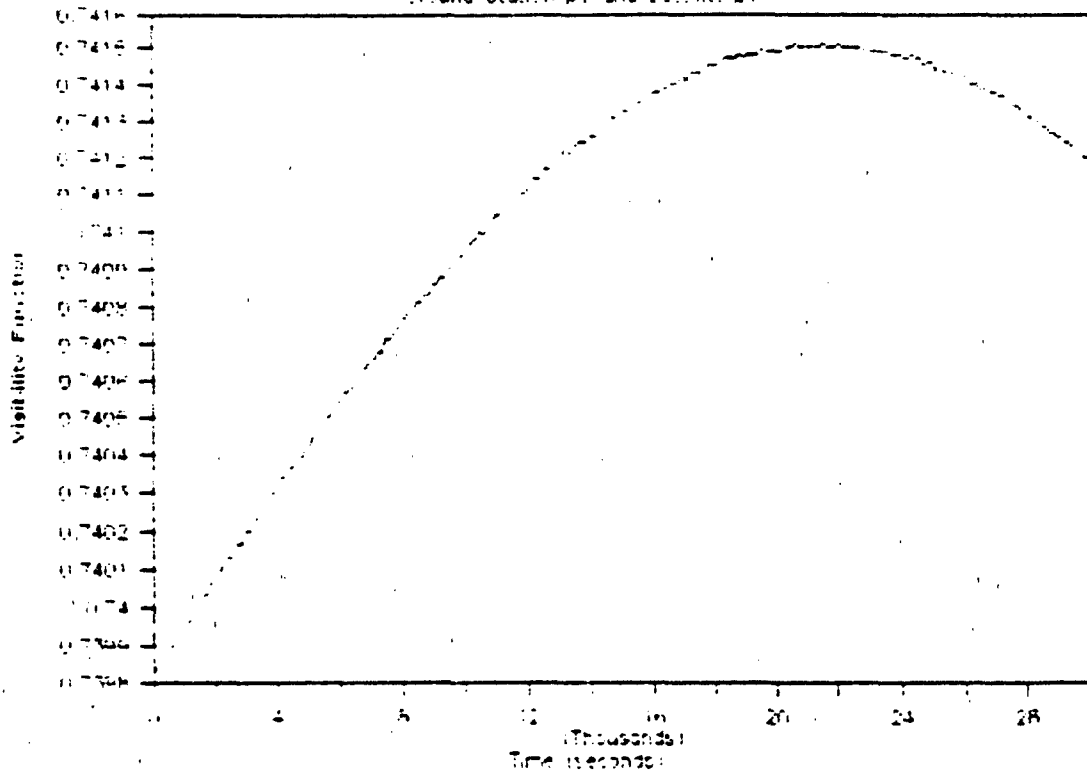
SATELLITE #4:

COMMUNICATIONS STATUS BETWEEN GROUND STATION #1 AND THE FOUR GIVEN SATELLITES:

(NOTE: NO ENTRIES OR MESSAGE BELOW INDICATES CONTINUOUS COMMUNICATIONS.)

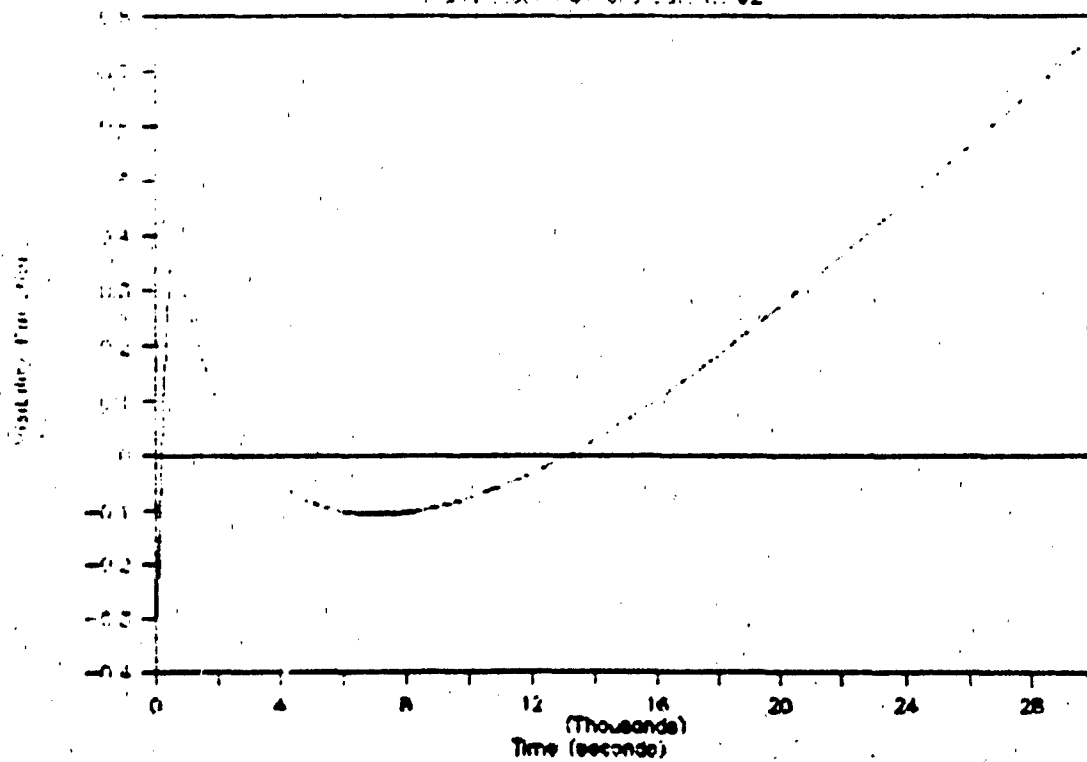
### VISIBILITY FUNCTION

Ground Station #1 and Satellite #1



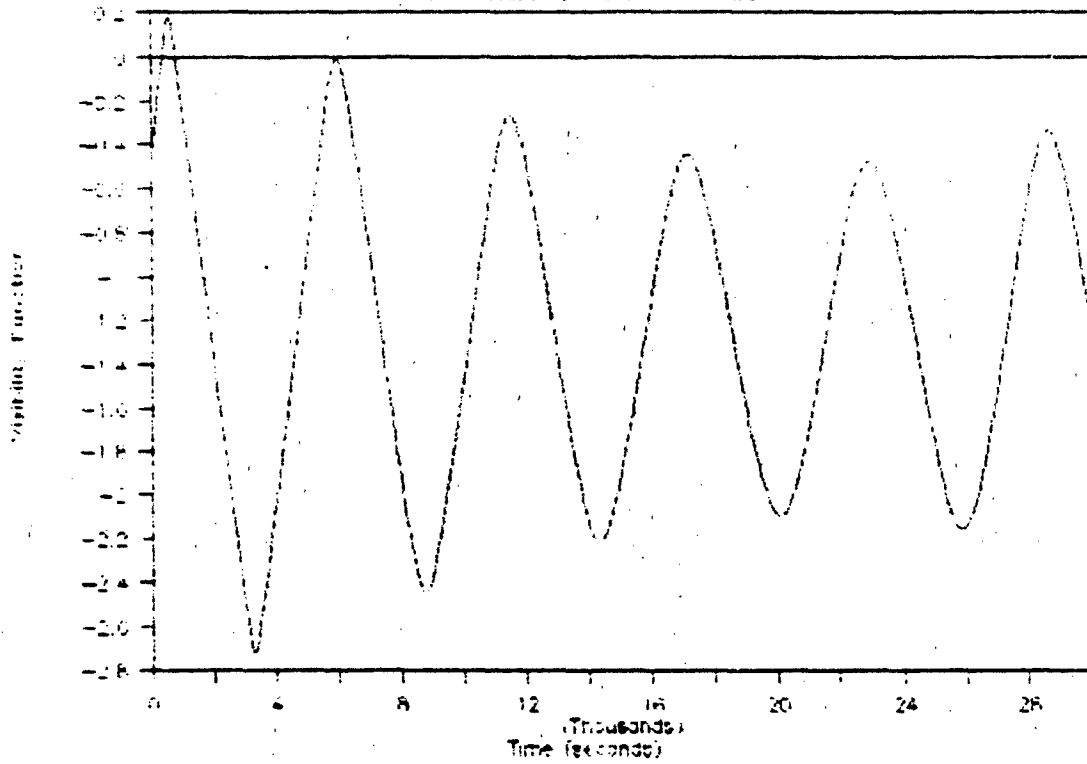
### VISIBILITY FUNCTION

Ground Station #1 and Satellite #2



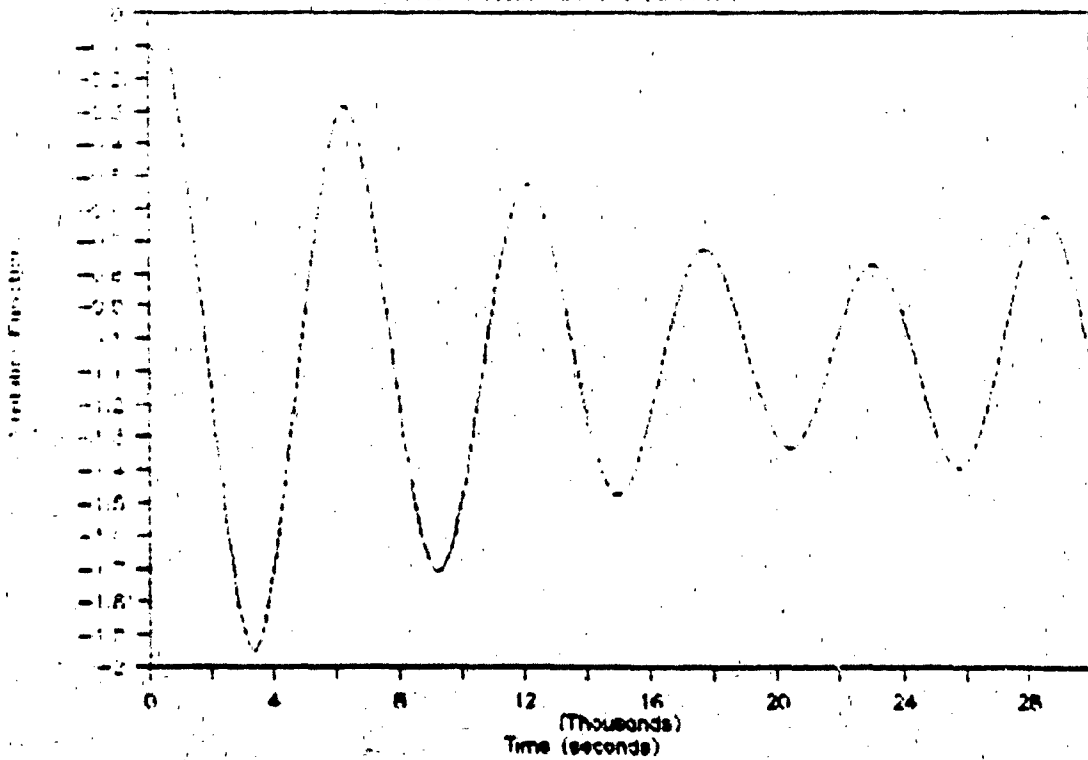
# VISIBILITY FUNCTION

Ground Station #1 and Satellite #3



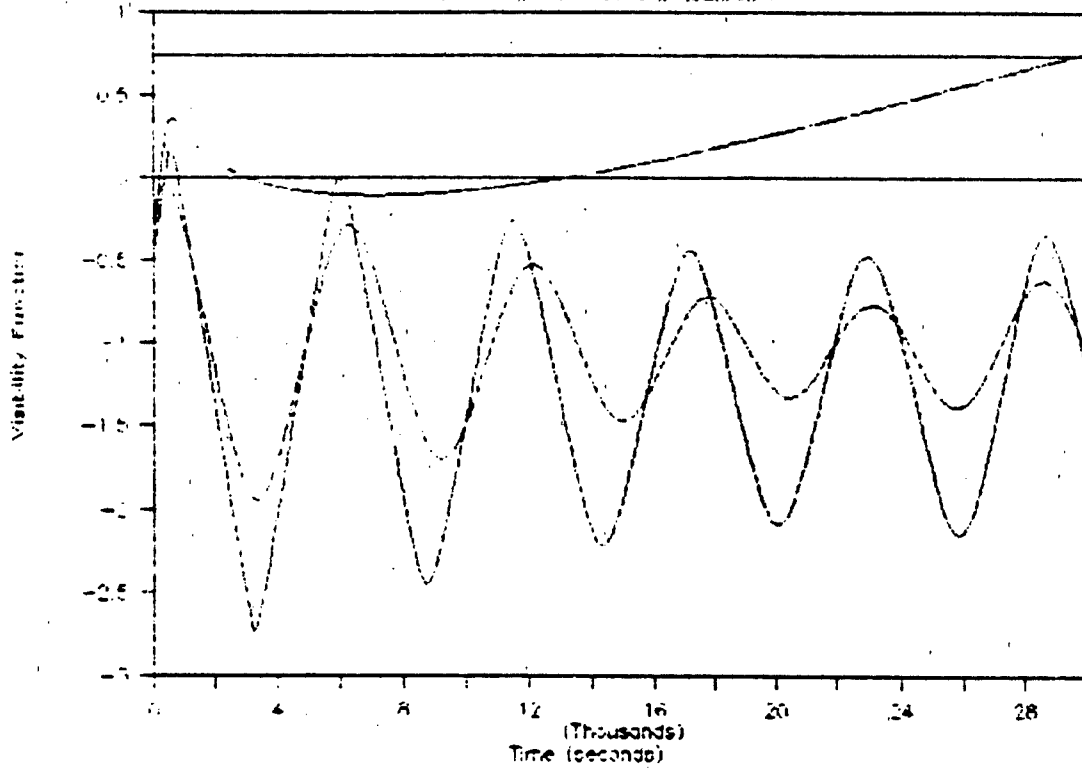
# VISIBILITY FUNCTION

Ground Station #1 and Satellite #2



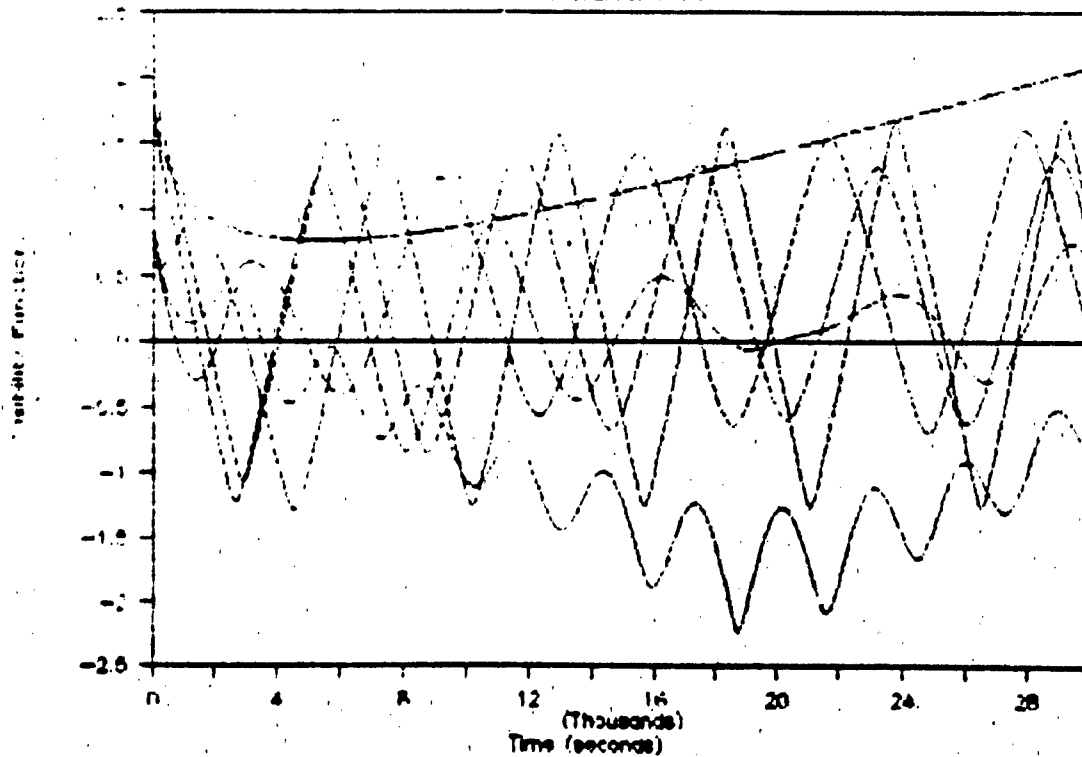
### VISIBILITY FUNCTION

Grid Station #1 and Sats #1, #2, #3, #4



### VISIBILITY FUNCTION

Satellites #1, #2, #3, and #4



Appendix C: Program Run #3

TIME FOR BEGINNING THE SIMULATION: 17:30.45

40

THE LOCATION OF GROUND STATION #1

EAST LONGITUDE: 201.735000  
GEODETIC LATITUDE: 21.571500  
ALTITUDE ABOVE THE EARTH ELLIPSOID: 0.000000

THE LOCATION OF GROUND STATION #2

EAST LONGITUDE: 55.480000  
GEODETIC LATITUDE: -4.669900  
ALTITUDE ABOVE THE EARTH ELLIPSOID: 0.000000

THE LOCATION OF GROUND STATION #3

EAST LONGITUDE: 291.400000  
GEODETIC LATITUDE: 76.508200  
ALTITUDE ABOVE THE EARTH ELLIPSOID: 0.000000

THE LOCATION OF GROUND STATION #4

EAST LONGITUDE: 239.498300  
GEODETIC LATITUDE: 34.823100  
ALTITUDE ABOVE THE EARTH ELLIPSOID: 0.000000

INITIAL ORBITAL ELEMENTS OF SATELLITE #1:

Eccentricity : 0.0668128000  
Semi-major Axis : 117819.33914850 KM  
Inclination : 57.35000000 DEG  
Mean Anomaly : 274.64810000 DEG  
Longitude of the Ascending Node : 65.63070000 DEG  
Argument of Periapsis : 79.11000000 DEG

INITIAL ORBITAL ELEMENTS OF SATELLITE #2:

Eccentricity : 0.0003109000  
Semi-major Axis : 42164.58832806 KM  
Inclination : 0.00990000 DEG  
Mean Anomaly : 132.80340000 DEG  
Longitude of the Ascending Node : 227.28640000 DEG  
Argument of Periapsis : 359.92590000 DEG

INITIAL ORBITAL ELEMENTS OF SATELLITE #3:

Eccentricity : 0.0145072000  
Semi-major Axis : 7496.14095441 KM  
Inclination : 90.26190000 DEG  
Mean Anomaly : 246.05610000 DEG  
Longitude of the Ascending Node : 107.70380000 DEG  
Argument of Periapsis : 115.56590000 DEG

INITIAL ORBITAL ELEMENTS OF SATELLITE #4:

Eccentricity : 0.0531098000  
Semi-major Axis : 7210.27162153 KM  
Inclination : 66.05630000 DEG  
Mean Anomaly : 245.56390000 DEG  
Longitude of the Ascending Node : 108.27480000 DEG  
Argument of Periapsis : 119.97980000 DEG



Rise/Set Times between Satellite #1 and Satellite #3:

2096.74276 SEC - SET  
3749.77870 SEC - RISE  
8657.22758 SEC - SET  
10396.52711 SEC - RISE  
15206.50054 SEC - SET  
17033.58950 SEC - RISE  
21747.79647 SEC - SET  
23658.24122 SEC - RISE  
28283.42317 SEC - SET

Rise/Set Times between Satellite #1 and Satellite #4:

1307.23847 SEC - SET  
3272.45823 SEC - RISE  
7507.23028 SEC - SET  
9472.56593 SEC - RISE  
13707.25247 SEC - SET  
15679.29431 SEC - RISE  
19906.64430 SEC - SET  
21891.90738 SEC - RISE  
26104.35521 SEC - SET  
28109.37531 SEC - RISE

Rise/Set Times between Satellite #2 and Satellite #3:

9589.46045 SEC - SET  
9952.64793 SEC - RISE  
15090.00549 SEC - SET  
17190.80137 SEC - RISE  
21406.02460 SEC - SET  
23759.34470 SEC - RISE  
27853.51599 SEC - SET

Rise/Set Times between Satellite #2 and Satellite #4:

146.81313 SEC - SET  
2048.36595 SEC - RISE  
7176.32283 SEC - SET  
9093.61752 SEC - RISE  
13570.46290 SEC - SET  
15790.10429 SEC - RISE  
19826.60084 SEC - SET  
22187.89728 SEC - RISE  
26069.57330 SEC - SET  
28469.81596 SEC - RISE

Rise/Set Times between Satellite #3 and Satellite #4:

13706.79928 SEC - SET  
16641.32575 SEC - RISE  
18745.74240 SEC - SET

COMMUNICATIONS STATUS BETWEEN SATELLITE #1, #2, #3, AND #4: (NOTE: NO ENTRIES OR MESSAGE BELOW INDICATES CONTINUOUS COMMUNICATIONS.)

Communications broken at 15090.00549 SEC  
Communications re-established at 15790.10429 SEC  
Communications broken at 21406.02460 SEC  
Communications re-established at 22187.89728 SEC  
Communications broken at 27853.51599 SEC  
Communications re-established at 28109.37531 SEC  
Communications broken at 28283.42317 SEC  
Communications re-established at 28469.81596 SEC

27189.14692 SEC - SET

42

RISE AND SET TIMES BETWEEN GROUND STATION #1 AND

SATELLITE #2:

RISE AND SET TIMES BETWEEN GROUND STATION #1 AND

SATELLITE #3:

6659.66079 SEC - RISE

7205.64965 SEC - SET

12742.74680 SEC - RISE

13876.25363 SEC - SET

19342.09950 SEC - RISE

20205.80586 SEC - SET

RISE AND SET TIMES BETWEEN GROUND STATION #1 AND

SATELLITE #4:

12011.00325 SEC - RISE

12888.92281 SEC - SET

18219.91633 SEC - RISE

19170.28556 SEC - SET

COMMUNICATIONS STATUS BETWEEN GROUND STATION #1 AND THE FOUR GIVEN SATELLITES:  
(NOTE: NO ENTRIES OR MESSAGE BELOW INDICATES CONTINUOUS COMMUNICATIONS.)

\*

RISE AND SET TIMES BETWEEN GROUND STATION #2 AND  
18845.80416 SEC - RISE

SATELLITE #1:

43

RISE AND SET TIMES BETWEEN GROUND STATION #2 AND

SATELLITE #2:

RISE AND SET TIMES BETWEEN GROUND STATION #2 AND

SATELLITE #3:

2776.71326 SEC - RISE

3809.60547 SEC - SET

9232.43874 SEC - RISE

10218.11934 SEC - SET

RISE AND SET TIMES BETWEEN GROUND STATION #2 AND

SATELLITE #4:

2575.55488 SEC - RISE

3233.90819 SEC - SET

8752.25542 SEC - RISE

9560.57871 SEC - SET

COMMUNICATIONS STATUS BETWEEN GROUND STATION #2 AND THE FOUR GIVEN SATELLITES:  
(NOTE: NO ENTRIES OR MESSAGE BELOW INDICATES CONTINUOUS COMMUNICATIONS.)

Communications broken at 0.000000000000000

Communications re-established at 2575.55488485389

Communications broken at 3809.60546671516

Communications re-established at 8752.25541685105

Communications broken at 10218.1193431246

Communications re-established at 18845.8041619931

\*

RISE AND SET TIMES BETWEEN GROUND STATION #3 AND

SATELLITE #1:

RISE AND SET TIMES BETWEEN GROUND STATION #3 AND

SATELLITE #2:

863.15409 SEC - RISE  
 1929.46674 SEC - SET  
 7439.15083 SEC - RISE  
 8459.80178 SEC - SET  
 14032.54055 SEC - RISE  
 15028.79748 SEC - SET  
 20602.42738 SEC - RISE  
 21618.43278 SEC - SET  
 27132.37091 SEC - RISE  
 28188.78822 SEC - SET

RISE AND SET TIMES BETWEEN GROUND STATION #3 AND

SATELLITE #3:

994.31344 SEC - RISE  
 1550.61962 SEC - SET  
 7128.82589 SEC - RISE  
 7759.49949 SEC - SET  
 13314.48646 SEC - RISE  
 13955.25665 SEC - SET  
 19524.67234 SEC - RISE  
 20130.86382 SEC - SET  
 25748.10806 SEC - RISE  
 26262.61775 SEC - SET

COMMUNICATIONS STATUS BETWEEN GROUND STATION #3 AND THE FOUR GIVEN SATELLITES:

(NOTE: NO ENTRIES OR MESSAGE BELOW INDICATES CONTINUOUS COMMUNICATIONS.)

Communications broken at 0.0000000000000000  
 Communications re-established at 863.154088411520  
 Communications broken at 1929.46674256308  
 Communications re-established at 7128.82589091173  
 Communications broken at 8459.80177798542  
 Communications re-established at 13314.4864571971  
 Communications broken at 13955.2566531969  
 Communications re-established at 14032.5405513214  
 Communications broken at 15028.7974810107  
 Communications re-established at 19524.6723377486  
 Communications broken at 20130.8638220675  
 Communications re-established at 20602.4273755799  
 Communications broken at 21618.4327784578  
 Communications re-established at 25748.1080570079  
 Communications broken at 26262.6177511727  
 Communications re-established at 27132.3709121300  
 Communications broken at 28188.7882164166

\*

RISE AND SET TIMES BETWEEN GROUND STATION #4 AND  
15864.12078 SEC - SET

SATELLITE #1:

45

RISE AND SET TIMES BETWEEN GROUND STATION #4 AND

SATELLITE #2:

RISE AND SET TIMES BETWEEN GROUND STATION #4 AND

SATELLITE #3:

160.74120 SEC - RISE  
1145.77540 SEC - SET  
6518.52238 SEC - RISE  
7628.58607 SEC - SET  
13356.74253 SEC - RISE  
13863.55253 SEC - SET

RISE AND SET TIMES BETWEEN GROUND STATION #4 AND

SATELLITE #4:

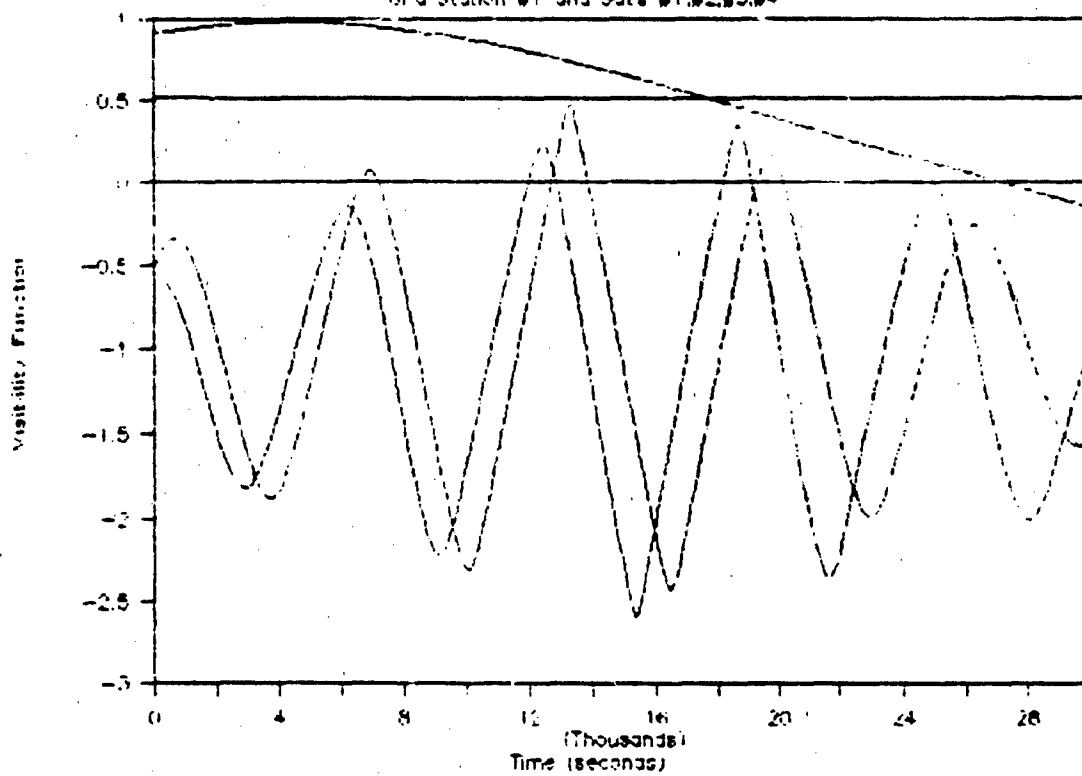
6188.38578 SEC - RISE  
7079.50299 SEC - SET  
12477.73425 SEC - RISE  
13319.74884 SEC - SET  
19231.12443 SEC - RISE  
19321.51031 SEC - SET

COMMUNICATIONS STATUS BETWEEN GROUND STATION #4 AND THE FOUR GIVEN SATELLITES:  
(NOTE: NO ENTRIES OR MESSAGE BELOW INDICATES CONTINUOUS COMMUNICATIONS.)

Communications broken at 15864.1207815275  
Communications re-established at 19231.1244274451  
Communications broken at 19321.5103139570

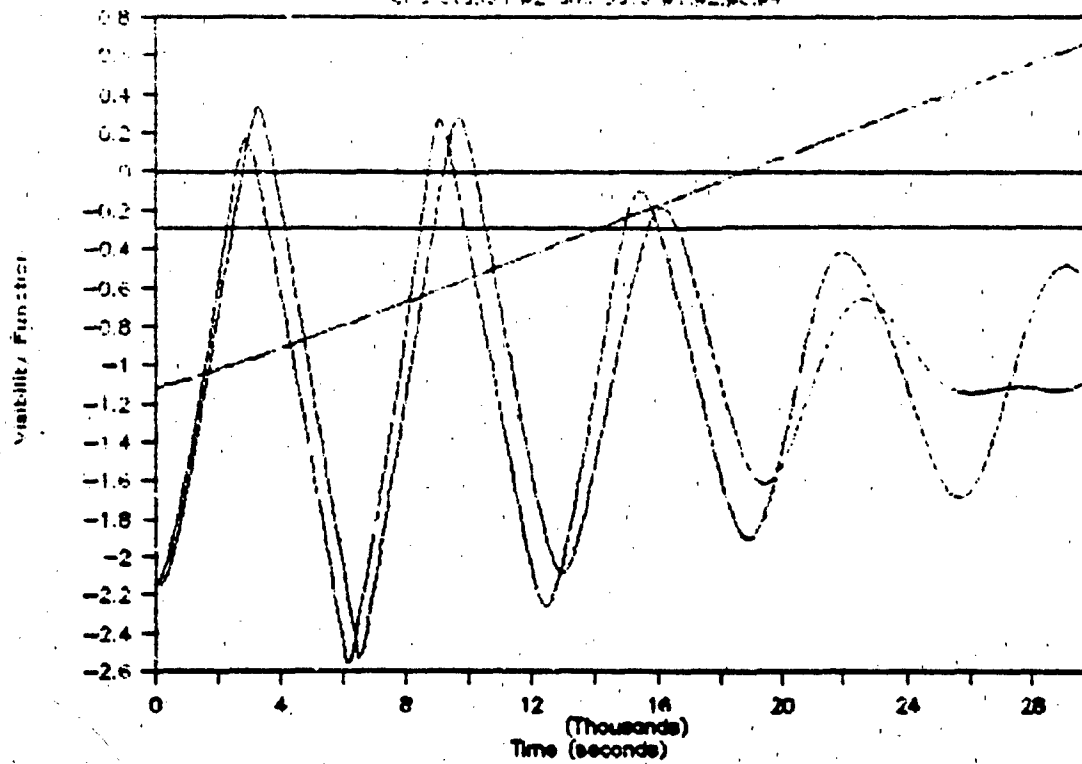
### VISIBILITY FUNCTION: DATA SET #1

Grid Station #1 and Sats #1, #2, #3, #4



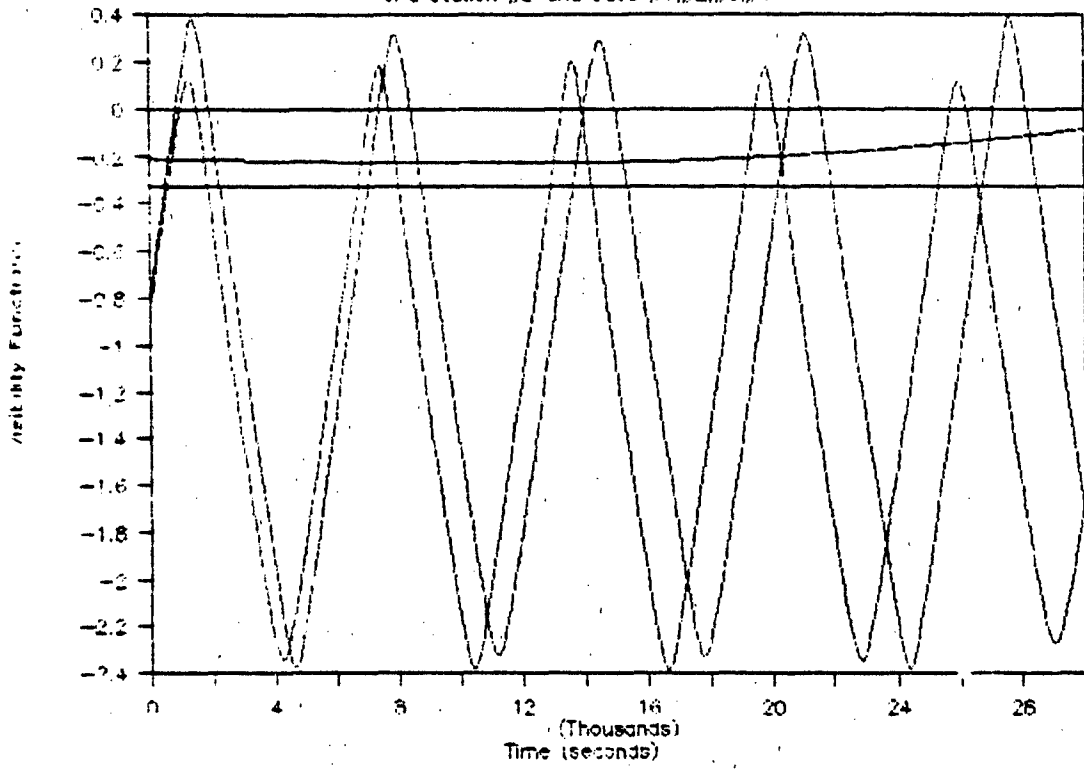
### VISIBILITY FUNCTION: DATA SET #2

Grid Station #2 and Sats #1, #2, #3, #4



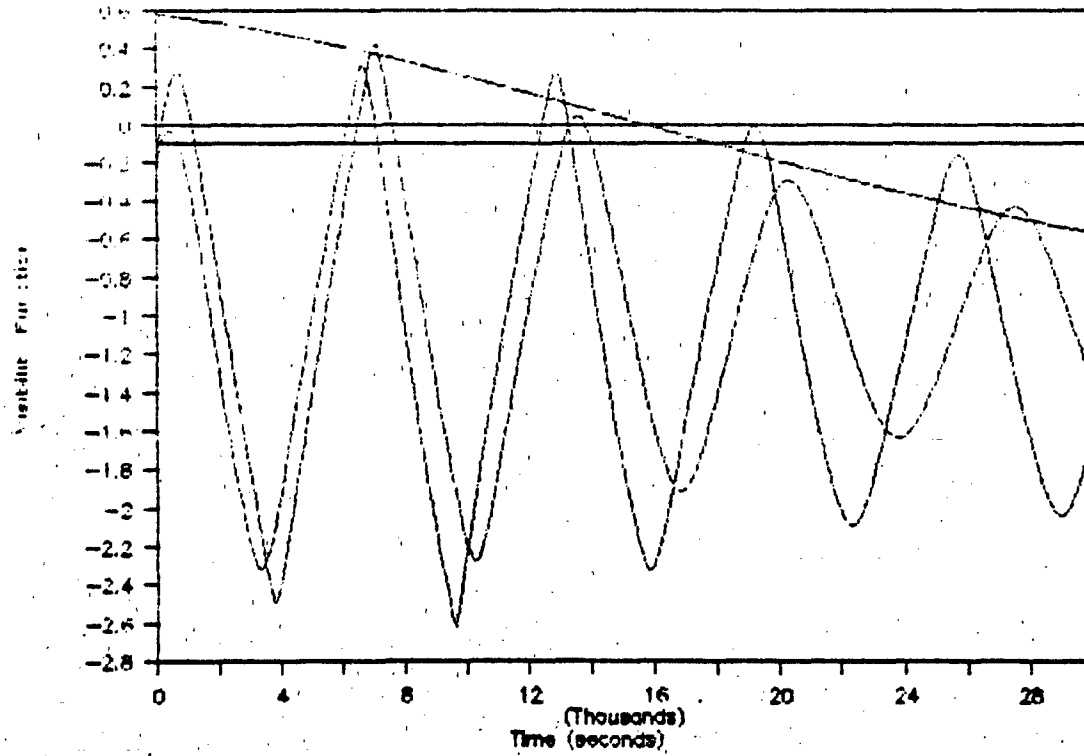
# VISIBILITY FUNCTION: DATA SET #2

Grid Station #3 and Sats #1,#2,#3,#4



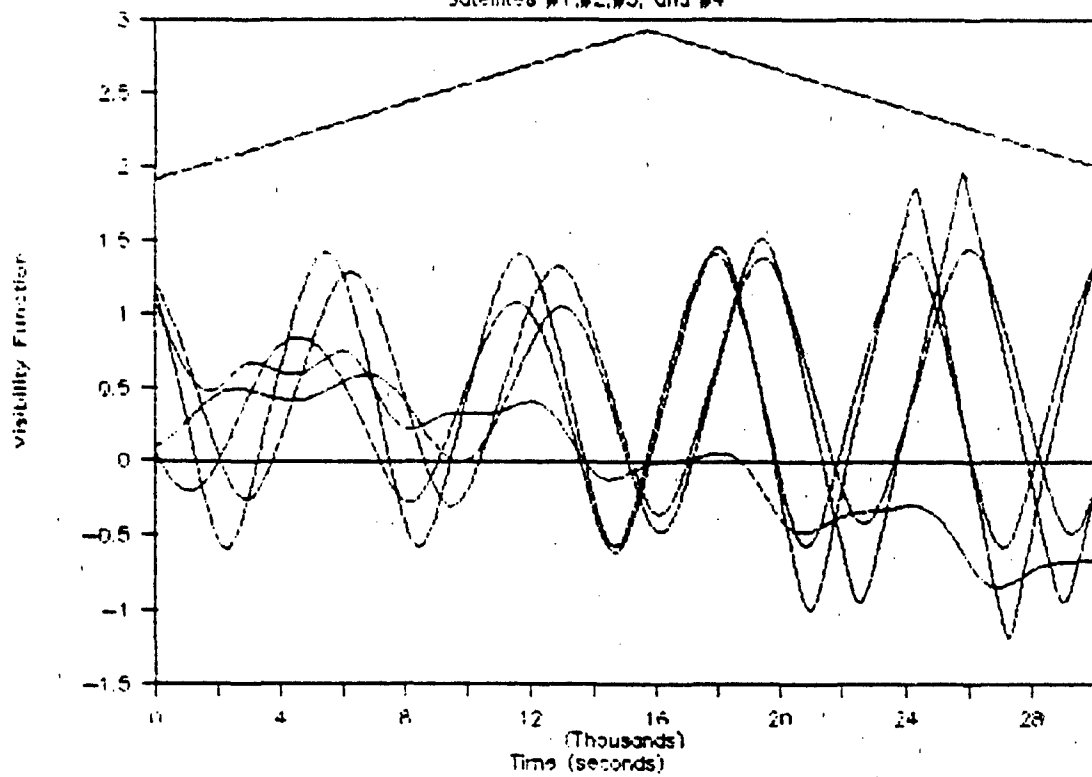
# VISIBILITY FUNCTION: DATA SET #2

Grid Station #4 and Sats #1,#2,#3,#4



# VISIBILITY FUNCTION: DATA SET #2

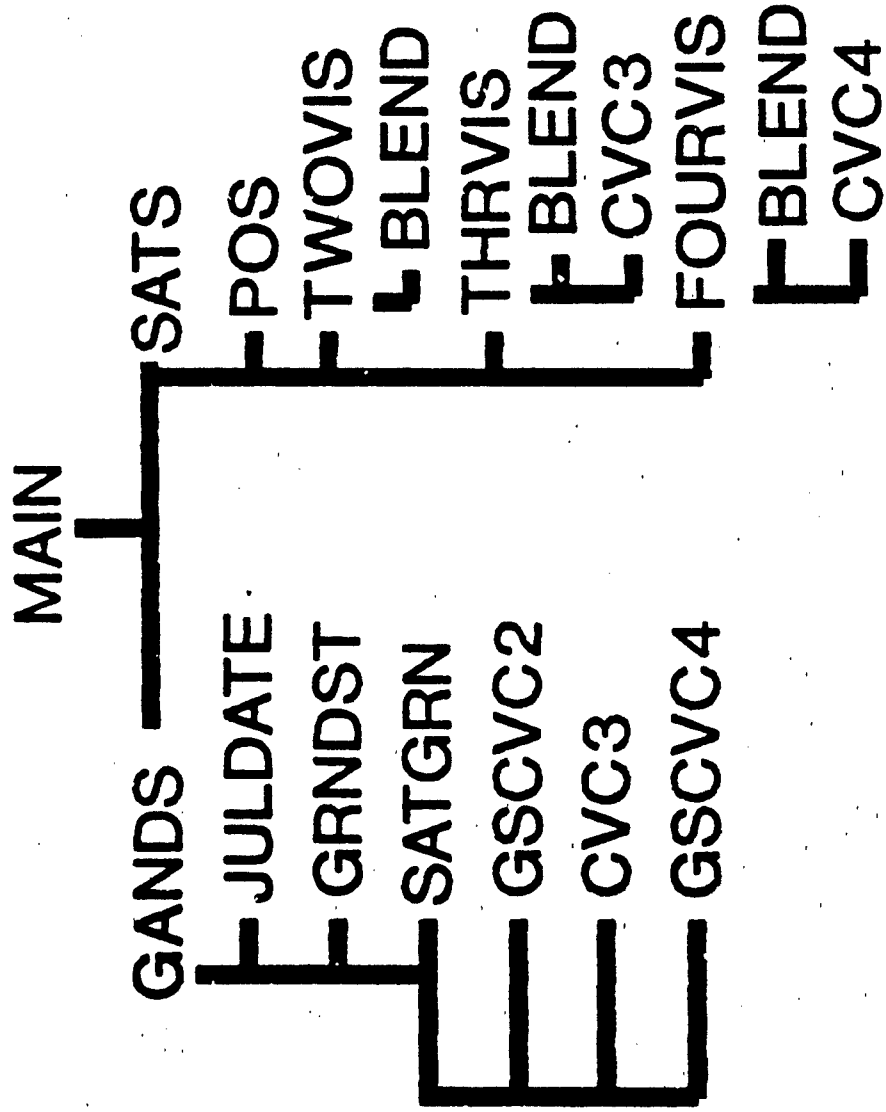
Satellites #1, #2, #3, and #4





Appendix D: Program Listing

# PROGRAM TREE



```

* PURPOSE: This program determines rise set times for inview *
* periods for satellite to satellite or satellite to *
* ground station viewing. This program is designed *
* to accommodate up to four satellites and as many *
* ground station. The following information is *
* necessary to be able to use the program: *
* Satellite Orbit: 1) Eccentricity *
*                  2) Length of Semi-major Axis *
*                  3) Inclination *
*                  4) Mean Anomaly *
*                  5) Longitude of the Ascending *
*                     Node *
*                  6) Argument of Periapsis *
* Ground Station: 1) Julian Date in Question *
*                  2) East Longitude *
*                  3) Height Above Sea Level *
* Other required information includes the amount of time *
* of interest and the desired time step for incrementa- *
* tion. The program is not limited to any particular *
* type of orbit. *changed to mean motion *

```

VARIABLES:

```

* CHOICE - the number of satellites indicated by the user *
* ELIP2 - square of the eccentricity of the earth *
* NUM - the number of ground stations indicated by the user *
* S2TU - converts SEC to TU; includes the steps size *
* SIZE - step size (SEC) *
* STEPS - number of steps *
* TIME - total time of interest (MIN) *
* YORN - yes or no response *
* SOROB - spherical or oblate earth *

```

```

*****
REAL*8 ELIP2,TIME,SIZE,S2TU
INTEGER CHOICE,NUM,SOROB,STEPS,YORN

```

```

OPEN(UNIT=25,FILE='RISESET.DAT',STATUS='NEW')

```

```

*****
* USERS CHOICE *
*****

```

```

WRITE(*,*) ' This program is designed to determine rise and'
WRITE(*,*) ' set times for satellite to satellite viewing as well'
WRITE(*,*) ' as satellite to ground station viewing. If you choose'
WRITE(*,*) ' to enter ground stations, satellite to satellite rise'
WRITE(*,*) ' and set times will be computed as well as satellite'
WRITE(*,*) ' to ground station rise and set times.'
WRITE(*,*)

```

```

1 WRITE(*,*) ' Are you interested in satellite to ground station'
WRITE(*,*) ' viewing? 1)Yes 2)No '
WRITE(*,*) ' =====> '
READ(*,*,ERR=1)YORN

```

```

IF (YORN.GT.2.OR.YORN.LT.1) THEN
WRITE(*,*) 'Your selection was not a valid response. Please'
WRITE(*,*) 'select again.'
GOTO 5
ENDIF

```

```

2 WRITE(*,*) 'Are you interested in modeling a spherical earth'
WRITE(*,*) 'or an oblate earth? 1)Spherical 2)Oblate'
WRITE(*,*) ' =====> '
READ(*,*,ERR=2)SOROB

```

```
ELSE IF (SOROB.EQ.2) THEN
  ELIP2 = 0.00669437999013D0
ENDIF
```

```
*****
* The following prompts the user to enter the values of the *
* time of interest and the step size. *
*****
```

```
5 WRITE(*,*)'What is the total time of interest (in minutes) ==> '
  READ(*,*,ERR=5)TIME
  WRITE(*,*)'The time entered above will be evaluated in steps.'
  WRITE(*,*)'The choice of step size is up to you; however, it '
  WRITE(*,*)'should be noted that step sizes too large may result'
  WRITE(*,*)'in missed rise and set times. A suggested step size'
  WRITE(*,*)'is 250 seconds.'
6 WRITE(*,*)'What is the step size of interest (in secs) ==> '
  READ(*,*,ERR=6)SIZE
```

```
TIME = TIME*60.0D0
STEPS = DINT(TIME/SIZE)
S2TU = SIZE/806.8103818D0
```

```
*****
* The following prompts the user to enter the number of ground *
* stations and/or satellites he/she desires *
*****
```

```
IF (YORN.EQ.1) THEN
8  WRITE(*,*)'How many ground stations do you desire. Please'
  WRITE(*,*)'make your selection from below.'
  WRITE(*,*)' 1) 1 Ground Station'
  WRITE(*,*)' 2) 2 Ground Stations'
  WRITE(*,*)' 3) 3 Ground Stations'
  WRITE(*,*)' 4) 4 Ground Stations'
  WRITE(*,*)'====> '
  READ(*,*,ERR=8)NUM
```

```
IF (NUM.LT.1.OR.NUM.GT.4) THEN
  WRITE(*,*)'Your entry was not valid. Please reenter the'
  WRITE(*,*)'number of ground stations.'
  GOTO 8
ENDIF
```

```
9  WRITE(*,*)'How many satellites do you wish to enter? Please'
  WRITE(*,*)'make your selection from below.'
  WRITE(*,*)' 1) 1 Satellite '
  WRITE(*,*)' 2) 2 Satellites'
  WRITE(*,*)' 3) 3 Satellites'
  WRITE(*,*)' 4) 4 Satellites'
  WRITE(*,*)'====> '
  READ(*,*,ERR=9)CHOICE
  CALL GANDS(CHOICE,ELIP2,NUM,S2TU,SIZE,STEPS)
```

```
ELSE
10 WRITE(*,*)'Please make a selection for the type of viewing'
  WRITE(*,*)'you prefer:'
  WRITE(*,*)' 1) 2 Satellite Viewing'
  WRITE(*,*)' 2) 3 Satellite Viewing'
  WRITE(*,*)' 3) 4 Satellite Viewing'
  WRITE(*,*)'====> '
  READ(*,*,ERR=10)CHOICE
  NUM = 0
```

```
IF (CHOICE.LT.1.OR.CHOICE.GT.3) THEN
  WRITE(*,*)'Your choice was not valid. Please re-enter'
```

-----  
CALL SATS(CHOICE,ELIP2,S2TU,SIZE,STEPS)

ENDIF

CLOSE(UNIT=10)

CLOSE(UNIT=11)

CLOSE(UNIT=12)

CLOSE(UNIT=13)

CLOSE(UNIT=14)

CLOSE(UNIT=15)

CLOSE(UNIT=25)

CLOSE(UNIT=31)

CLOSE(UNIT=32)

CLOSE(UNIT=33)

CLOSE(UNIT=34)

END

\*

```

*
* PURPOSE: The purpose of this subroutine is to coordinate the
*           determination of rise/set times between satellites
*           and ground stations.
*
* VARIABLES:
* INPUT:
* CHOICE - the number of satellites indicated by the user
* NUM    - the number of ground stations indicated by the user
* S2TU   - converts SEC to TU
* SIZE   - step size (SEC)
* STEPS  - number of steps
*
* LOCAL:
* DAY    - 1 to 31
* ELIP2  - square of the eccentricity of the earth
* ELONG  - east longitude position of the ground station
* GLAT   - geodetic latitude of the ground station
* H      - altitude above the earth ellipsoid
* HOUR   - 0 to 23
* JD     - Julian Date
* KIND   - determines whether continuous visibility involves
*           a ground station and three satellites (0) or three
*           satellites (1)
* MIN    - 0 to 59
* MON    - 1 to 12
* SEC    - 0 to 59
* STATE  - variable that defines file number
* T      - time corresponding to a given step
* VAR    - variable that defines file number
* VIS    - value of the visibility function at a given step
* YEAR   - 1900 to 2100

```

```

*****

```

```

SUBROUTINE GANDS(CHOICE,ELIP2,NUM,S2TU,SIZE,STEPS)

```

```

REAL*8 DAY,ELIP2,ELONG,GLAT,H,HOUR,JD,MIN,MON,S2TU
REAL*8 SEC,SIZE,T(0:2005).VIS(0:2005).YEAR
INTEGER CHOICE,COUNT,IN,INTURN,KIND,NUM.OUT,STATE,STEPS,VAR

```

```

OPEN(UNIT=31,FILE='GSPOS1.DAT',STATUS='NEW')
OPEN(UNIT=32,FILE='GSPOS2.DAT',STATUS='NEW')
OPEN(UNIT=33,FILE='GSPOS3.DAT',STATUS='NEW')
OPEN(UNIT=34,FILE='GSPOS4.DAT',STATUS='NEW')

```

```

*****

```

```

* In order to accurately determine the position vectors of the
* given ground station, it is necessary to know the exact time
* (Universal Time) the simulation is to start. The following
* lines prompt the user to enter the year, month, day, hour, min-
* ute, and second the simulation is to start. The Julian Date
* will be calculated from this information.

```

```

*****

```

```

WRITE(*,*)'In order to determine position vectors of the earth'
WRITE(*,*)'ground station(s), it is necessary to know the year,'
WRITE(*,*)'month, day, and time when you wish the simulation to
WRITE(*,*)'begin. The time of interest is Universal Time.'
WRITE(*,*)'Please enter the following information:'

```

```

*****

```

```

YEAR

```

```

*****

```

```

5 WRITE(*,*)' What is the year of interest, between 1900 and 2100?'
WRITE(*,190)
READ(*,*.ERR=5)YEAR

```

```

GOTO 5
ENDIF
*****
*                               MONTH                               *
*****
10 WRITE(*,*) ' What is the month of interest? Please enter the '
WRITE(*,*) ' number of the month between 1 and 12.'
WRITE(*,190)
READ(*,*,ERR=10)MON
IF (MON.LT.1.OR.MON.GT.12) THEN
WRITE(*,*) ' The month you entered was not a valid response.'
WRITE(*,*) ' Please re-enter the value.'
GOTO 10
ENDIF
*****
*                               DAY                               *
*****
15 WRITE(*,*) ' What is the day of interest? Please enter a value'
WRITE(*,*) ' between 1 and 31.'
WRITE(*,190)
READ(*,*,ERR=15)DAY
IF (DAY.LT.1.OR.DAY.GT.31) THEN
WRITE(*,*) ' The day you entered was not valid. Please'
WRITE(*,*) ' re-enter the value'
GOTO 15
ELSE IF (MON.EQ.2.AND.DAY.GT.29) THEN
WRITE(*,*) 'February does not have more than 29 days. Please'
WRITE(*,*) 'reenter the month.'
GOTO 15
ELSE IF (DAY.EQ.31) THEN
IF (MON.EQ.4.OR.MON.EQ.6.OR.MON.EQ.9.OR.MON.EQ.11) THEN
WRITE(*,*) ' The month you entered does not have more than 30'
WRITE(*,*) ' days. Please re-enter the month.'
GOTO 15
ENDIF
ENDIF
*****
*                               HOUR                               *
*****
WRITE(*,*) 'Next it is necessary to enter the time of interest.'
WRITE(*,*) 'This will be done by hour, minute, and second.'
WRITE(*,*) 'Please respond at the prompt.'
WRITE(*,*)
20 WRITE(*,*) ' What is the hour you wish the simulation to begin?'
WRITE(*,*) ' The value must be between 1 and 23.'
WRITE(*,190)
READ(*,*,ERR=20)HOUR
IF (HOUR.LT.0.OR.HOUR.GT.23) THEN
WRITE(*,*) ' The hour you entered was not valid. Please '
WRITE(*,*) ' re-enter the hour.'
GOTO 20
ENDIF
*****
*                               MINUTE                               *
*****
25 WRITE(*,*) ' At what minute in the hour do you wish to begin.'
WRITE(*,*) ' Please enter a number between 1 and 59.'
WRITE(*,190)
READ(*,*,ERR=25)MIN
IF (MIN.LT.0.OR.MIN.GT.59) THEN
WRITE(*,*) ' The minute you entered was not a valid response.'
WRITE(*,*) ' Please re-enter the value.'
GOTO 25

```

```

-----*****
30 WRITE(*,*)' At what second do you wish to begin. Please enter'
   WRITE(*,*)' a number between 1 and 59.'
   WRITE(*,190)
   READ(*,*,ERR=30)SEC
   IF (SEC.LT.0.OR.SEC.GT.59) THEN
     WRITE(*,*)' The second you entered was not a valid response.'
     WRITE(*,*)' Please re-enter the value.'
     GOTO 30
   ENDIF

   WRITE(25,150)MON,DAY,YEAR
   WRITE(25,160)HOUR,MIN,SEC
   WRITE(25,*)

*****
* The following line calls the subroutine JULDATE where the Julian *
* Date is calculated. *
*****
   CALL JULDATE(DAY,HOUR,MIN,MON,SEC,YEAR,JD)
*****
* The following DO loop call for the user to enter the east *
* longitude and the geodetic latitude. The ground station position *
* vectors are then determined in subroutine GRNDST. *
*****
   DO 40 COUNT = 1,NUM
33  WRITE(*,*)'Please enter the east longitude value for the '
     WRITE(*,200)COUNT
     WRITE(*,190)
     READ(*,*,ERR=33)ELONG
     IF (ELONG.LT.-360.000.OR.ELONG.GT.360.000) THEN
       WRITE(*,*)'The value of longitude entered was not valid.'
       WRITE(*,*)'Please re-enter the value.'
       GOTO 33
     ENDIF
     WRITE(25,220)COUNT
     WRITE(25,230)ELONG
35  WRITE(*,*)'Please enter the geodetic latitude for the '
     WRITE(*,200)COUNT
     WRITE(*,*)'The value should be between -90 and 90 degrees.'
     WRITE(*,190)
     READ(*,*,ERR=35)GLAT

     IF (GLAT.LT.(-90.000).OR.GLAT.GT.(90.000)) THEN
       WRITE(*,*)'The geodetic latitude value you entered was not'
       WRITE(*,*)'acceptable. Please re-enter the value.'
       GOTO 35
     ENDIF
     WRITE(25,240)GLAT

38  WRITE(*,*)'Please enter the altitude of the ground station '
     WRITE(*,*)'above sea level (in KM).'
     WRITE(*,190)
     READ(*,*,ERR=38)H
     WRITE(25,250)H
     WRITE(25,*)

*****
* Next, the subroutine GRNDST is call where the position vectors of *
* ground station are determined. *
*****
   CALL GRNDST(COUNT,ELIP2,ELONG,GLAT,H,JD,S2TU,STEPS)
40 CONTINUE
   CLOSE(UNIT=31)
   CLOSE(UNIT=32)

```



\* the subroutine SATS is called in order to determine the rise set \*  
\* times between the satellites entered in this portion of the \*  
\* program. \*

\*\*\*\*\*

```
CHOICE = CHOICE - 1
CALL SATS(CHOICE,ELIP2,S2TU,SIZE,STEPS)
CHOICE = CHOICE + 1
```

\*\*\*\*\*

\* The following opens the necessary files, then records the vis- \*  
\* ibility data to send to the subroutine BLEND. \*

\*\*\*\*\*

```
DO 70 OUT = 1,NUM
OPEN(UNIT=60,FILE='TRACKER.DAT',STATUS='UNKNOWN')
OPEN(UNIT=41,FILE='STAT1.DAT',STATUS='NEW')
OPEN(UNIT=42,FILE='STAT2.DAT',STATUS='NEW')
OPEN(UNIT=43,FILE='STAT3.DAT',STATUS='NEW')
OPEN(UNIT=44,FILE='STAT4.DAT',STATUS='NEW')
```

```
VAR = 1
```

```
DO 60 IN = 1,CHOICE
IF (OUT.EQ.1.AND.IN.EQ.1) THEN
OPEN(UNIT=31,FILE='GSPOS1.DAT',STATUS='OLD')
OPEN(UNIT=1,FILE='POS1.DAT',STATUS='OLD')
ELSE IF (OUT.EQ.1.AND.IN.EQ.2) THEN
OPEN(UNIT=31,FILE='GSPOS1.DAT',STATUS='OLD')
OPEN(UNIT=2,FILE='POS2.DAT',STATUS='OLD')
ELSE IF (OUT.EQ.1.AND.IN.EQ.3) THEN
OPEN(UNIT=31,FILE='GSPOS1.DAT',STATUS='OLD')
OPEN(UNIT=3,FILE='POS3.DAT',STATUS='OLD')
ELSE IF (OUT.EQ.1.AND.IN.EQ.4) THEN
OPEN(UNIT=31,FILE='GSPOS1.DAT',STATUS='OLD')
OPEN(UNIT=4,FILE='POS4.DAT',STATUS='OLD')

ELSE IF (OUT.EQ.2.AND.IN.EQ.1) THEN
OPEN(UNIT=32,FILE='GSPOS2.DAT',STATUS='OLD')
OPEN(UNIT=1,FILE='POS1.DAT',STATUS='OLD')
ELSE IF (OUT.EQ.2.AND.IN.EQ.2) THEN
OPEN(UNIT=32,FILE='GSPOS2.DAT',STATUS='OLD')
OPEN(UNIT=2,FILE='POS2.DAT',STATUS='OLD')
ELSE IF (OUT.EQ.2.AND.IN.EQ.3) THEN
OPEN(UNIT=32,FILE='GSPOS2.DAT',STATUS='OLD')
OPEN(UNIT=3,FILE='POS3.DAT',STATUS='OLD')
ELSE IF (OUT.EQ.2.AND.IN.EQ.4) THEN
OPEN(UNIT=32,FILE='GSPOS2.DAT',STATUS='OLD')
OPEN(UNIT=4,FILE='POS4.DAT',STATUS='OLD')

ELSE IF (OUT.EQ.3.AND.IN.EQ.1) THEN
OPEN(UNIT=33,FILE='GSPOS3.DAT',STATUS='OLD')
OPEN(UNIT=1,FILE='POS1.DAT',STATUS='OLD')
ELSE IF (OUT.EQ.3.AND.IN.EQ.2) THEN
OPEN(UNIT=33,FILE='GSPOS3.DAT',STATUS='OLD')
OPEN(UNIT=2,FILE='POS2.DAT',STATUS='OLD')
ELSE IF (OUT.EQ.3.AND.IN.EQ.3) THEN
OPEN(UNIT=33,FILE='GSPOS3.DAT',STATUS='OLD')
OPEN(UNIT=3,FILE='POS3.DAT',STATUS='OLD')
ELSE IF (OUT.EQ.3.AND.IN.EQ.4) THEN
OPEN(UNIT=33,FILE='GSPOS3.DAT',STATUS='OLD')
OPEN(UNIT=4,FILE='POS4.DAT',STATUS='OLD')

ELSE IF (OUT.EQ.4.AND.IN.EQ.1) THEN
OPEN(UNIT=34,FILE='GSPOS4.DAT',STATUS='OLD')
OPEN(UNIT=1,FILE='POS1.DAT',STATUS='OLD')
```

```
ELSE IF (OUT.EQ.4.AND.IN.EQ.3) THEN
  OPEN(UNIT=34,FILE='GSPOS4.DAT',STATUS='OLD')
  OPEN(UNIT=3,FILE='POS3.DAT',STATUS='OLD')
ELSE IF (OUT.EQ.4.AND.IN.EQ.4) THEN
  OPEN(UNIT=34,FILE='GSPOS4.DAT',STATUS='OLD')
  OPEN(UNIT=4,FILE='POS4.DAT',STATUS='OLD')
ENDIF
```

```
CALL SATGRN(IN,OUT,SIZE,STEPS)
CLOSE(UNIT=OUT+30)
CLOSE(UNIT=IN)
```

```
OPEN(UNIT=40,FILE='GSVIS.DAT',STATUS='OLD')
DO 50 INTURN = 1,STEPS+1
  READ(40,*)T(INTURN),VIS(INTURN)
  IF (IN.EQ.1) THEN
    OPEN(UNIT=10,FILE='VIS1.DAT',STATUS='UNKNOWN')
    WRITE(10,*)T(INTURN),VIS(INTURN)
    CLOSE(UNIT=10)
  ELSE IF (IN.EQ.2) THEN
    OPEN(UNIT=11,FILE='VIS2.DAT',STATUS='UNKNOWN')
    WRITE(11,*)T(INTURN),VIS(INTURN)
    CLOSE(UNIT=11)
  ELSE IF (IN.EQ.3) THEN
    OPEN(UNIT=12,FILE='VIS3.DAT',STATUS='UNKNOWN')
    WRITE(12,*)T(INTURN),VIS(INTURN)
    CLOSE(UNIT=12)
  ELSE IF (IN.EQ.4) THEN
    OPEN(UNIT=13,FILE='VIS4.DAT',STATUS='UNKNOWN')
    WRITE(13,*)T(INTURN),VIS(INTURN)
    CLOSE(UNIT=13)
```

```
50      ENDIF
      CONTINUE
```

```
WRITE(25,*)
WRITE(25,260)OUT,IN
CLOSE(UNIT = 40)
```

```
STATE = VAR+40
CALL BLEND(STATE,STEPS,T,VIS)
CLOSE(UNIT=STATE)
VAR=VAR+1
```

```
60 CONTINUE
IF (CHOICE.EQ.2) THEN
  WRITE(25,*)
  WRITE(25,265)OUT
  WRITE(25,285)
  STATE = 41
  CALL GSSCVC2(STATE)
ELSE IF (CHOICE.EQ.3) THEN
  WRITE(25,*)
  WRITE(25,270)OUT
  WRITE(25,285)
  KIND = 0
  STATE = 41
  CALL CVC3(KIND,STATE)
ELSE IF (CHOICE.EQ.4) THEN
  WRITE(25,*)
  WRITE(25,280)OUT
  WRITE(25,285)
  STATE = 41
  CALL GSSCVC4(STATE)
```

70 CONTINUE

RETURN

```
150 FORMAT('DATE FOR BEGINNING THE SIMULATION: ',I2,'/',I2,'/',I1)
160 FORMAT('TIME FOR BEGINNING THE SIMULATION: ',I2,':',I2,':',I2)
190 FORMAT(2X,'====> ')
200 FORMAT(' location of ground station #',I1,'.')
220 FORMAT('THE LOCATION OF GROUND STATION #',I1)
230 FORMAT('EAST LONGITUDE: ',F12.6)
240 FORMAT('GEODETTIC LATITUDE: ',F12.6)
250 FORMAT('ALTITUDE ABOVE THE EARTH ELLIPSOID: ',F12.6)
260 FORMAT('RISE AND SET TIMES BETWEEN GROUND STATION #',I1,' AND
+ SATELLITE #',I1,':')
265 FORMAT('COMMUNICATIONS STATUS BETWEEN GROUND STATION #',I1,' AND
+ THE TWO GIVEN SATELLITES:')
270 FORMAT('COMMUNICATIONS STATUS BETWEEN GROUND STATION #',I1,' AND
+ THE THREE GIVEN SATELLITES:')
280 FORMAT('COMMUNICATIONS STATUS BETWEEN GROUND STATION #',I1,' AND
+ THE FOUR GIVEN SATELLITES:')
285 FORMAT('(NOTE: NO ENTRIES OR MESSAGE BELOW INDICATES CONTINUOUS
+ COMMUNICATIONS.))')
END
```

\*

```

*
* PURPOSE: The purpose of this subroutine is to determine the
* visibility function between a given satellite and a
* given ground station.
*
* VARIABLES:
* INPUT:
*   IN      - do loop counter representing the number of sat's
*   OUT     - do loop counter representing the number of ground
*            stations
*   SIZE    - step size
*   STEPS   - total number of steps
*
* LOCAL:
*   COUNT   - do loop counter
*   PHI     - angle between the position vector to the satellite
*            and the position vector to the ground station
*   R1      - position vector to the ground station
*   R2      - position vector to the satellite
*   VIS     - value of the visibility function between satellite
*            and ground station
*
* REFERENCES: Lawson, 32
*             Alfano, Negron, and Moore, 2
*****
SUBROUTINE SATGRN(IN,OUT,SIZE,STEPS)

REAL*8 PHI,R1(4),R2(4),SIZE,VIS
INTEGER COUNT,IN,OUT,STEPS
*****
* GSVIS.DAT contains visibility data between the given satellite and *
* ground station. This file will be overwritten each time SATGRN is *
* called.
*****
OPEN(UNIT=40,FILE='GSVIS.DAT',STATUS='UNKNOWN')
DO 20 COUNT = 0,STEPS

    READ(OUT+30,*)R1(1),R1(2),R1(3),R1(4)
    READ(IN,*)R2(1),R2(2),R2(3),R2(4)
*****
* The next lines define the visibility function between the satellite*
* and the ground station.
* Reference: Lt Col Salvatore Alfano
*****
    PHI = ((R1(1)*R2(1))+R1(2)*R2(2))+R1(3)*R2(3))/(R1(4)*R2(4))

    IF (PHI.GT.(1.000)) THEN
        PHI = 1.000
    ELSE IF (PHI.LT.(-1.000)) THEN
        PHI = -1.000
    ENDIF

    VIS = DACOS((R1(4))/R2(4)) - DACOS(PHI)

    WRITE(40,110)COUNT*SIZE,VIS

20 CONTINUE

CLOSE(UNIT=40)

RETURN

```

110 FORMAT(I10,5X,F12.5)

END

\*

SUBROUTINE JULDATE

PURPOSE: The purpose of this subroutine is to convert conventional time to Julian Date.

VARIABLES:

INPUT:

DAY - 1 to 31  
HR - 0 to 23  
MIN - 0 to 59  
MON - 1 to 12  
SEC - 0 to 59  
YEAR - 1900 to 2100

OUTPUT:

JD - Julian Date

Reference: Valado, 1

SUBROUTINE JULDATE(DAY,HOUR,MIN,MON,SEC,YEAR,JD)

REAL\*8 DAY,HOUR,MIN,MON,SEC,YEAR,JD

JD = 367.000\*(YEAR)-DINT((7.000\*(YEAR+DINT((MON+9.000)/12.000)))  
+ /4.000)+DINT((275.000\*MON)/9.000)+DAY+1721013.500+  
+ (((SEC/60.000)+MIN)/60.000)+HOUR)/24.000

RETURN  
END

SUBROUTINE GRNDST

PURPOSE: The purpose of this subroutine is to determine the geocentric position of a ground station given Julian Date, east longitude, and geodetic latitude.

VARIABLES:

INPUT:

- ELIP2 - square of the eccentricity of the earth
- ELONG - east longitude of the station of interest
- GLAT - geodetic longitude of the station of interest
- H - altitude of the station above the earth ellipsoid
- JD - Julian Date
- S2TU - converts SEC to TU
- STEPS - number of steps

LOCAL VARIABLES:

- COSLAT - cosine of the geodetic latitude
- DEGRAD - conversion from degrees to radians
- DENOM - denominator of the equation to determine G1 and G2
- DT - change in time
- GLAT - geodetic latitude of the ground station
- GSR(1-4) - position vectors of the ground station
- KM2DU - conversion from km to DU
- REP - counter for do loop
- SINLAT - sine of the geodetic latitude
- TGO - Greenwich Sidereal Time at 0 hr Universal Time
- THETA - observers local sidereal time
- THETDOT - time rate of change of the local sidereal time based on the rotation of the earth (rad/TU)
- THETG - sidereal time of the Greenwich Meridian
- TU - time measured in centuries
- TWOPI - two times pi
- X - variable for the determination of station position vector

Reference: Valado, 2-4  
 Bates, Mueller, and White, 98-103  
 Escobal, 20-21  
 Alfano, Negrón, and Moore, 6

SUBROUTINE GRNDST(COUNT,ELIP2,ELONG,GLAT,H,JD,S2TU,STEPS)

REAL\*8 COSLAT,DEGRAD,DENOM,DT,ELIP2,ELONG,GLAT  
 REAL\*8 GSR(4),H,JD,KM2DU,S2TU,SINLAT,TGO  
 REAL\*8 THETA,THETDOT,THETG,TU,TWOPI,X  
 INTEGER COUNT,REP,STEPS

CONSTANTS

TWOPI = 2.000\*DACOS(-1.000)  
 DEGRAD = TWOPI/360.000  
 KM2DU = (1.000/6378.13700)

ELONG = ELONG\*DEGRAD  
 GLAT = GLAT\*DEGRAD  
 H = H\*KM2DU

THETDOT = .05883359068878600

The following lines calculate the Greenwich Sidereal Time and to be used in the determination of the Local Sidereal Time.

```
TU      = (DINT(JD) + .5D0 - 2451545.0D0)/36525.0D0
TGO     = 1.753368559D0+628.3319705D0*TU+
+       .0C0006770708127D0*(TU**2.0D0)+(6.30038809866571)*
+       DBLE(JD-DINT(JD)-.5D0)
```

```
TGO     = DMOD(TGO,TWOPI)
IF (TGO.LT.(0.0D0)) THEN
  TGO = TWOPI+TGO
ENDIF
```

```
DENOM   = DSQRT(1.0D0-(ELIP2*(SINLAT**2.0D0)))
```

```
X       = ((1.0D0/DENOM)+H)*COSLAT
GSR(3)  = (((1.0D0-ELIP2)/DENOM)+H)*SINLAT*(1.0D0/
+       DSQRT(1.0D0-ELIP2))
GSR(4)  = DSQRT((X**2.0D0)+(GSR(3)**2.0D0))
```

```
TGO = 0.0D0
```

```
*****
* The following lines calculate the x and z variables for the *
* position of the ground station. These variables are then used *
* to calculate the position vector in the IJK or geocentric coor- *
* dinate system. *
*****
```

```
DO 50 REP = 0,STEPS
  DT   = REP*S2TU
  THETG = TGO + THETDOT*DT
  THETA = THETG + ELONG
```

```
GSR(1) = X*DCOS(THETA)
GSR(2) = X*DSIN(THETA)
```

```
WRITE(COUNT+30,100)GSR(1),GSR(2),GSR(3),GSR(4)
```

```
50 CONTINUE
RETURN
```

```
100 FORMAT(F12.5,5X,F12.5,5X,F12.5,5X,F12.5)
```

```
END
```



```

*
*                               SUBROUTINE SATS                               *
*
* PURPOSE:
*
* VARIABLES:
* ORBITAL ELEMENTS:
*   ARGP - argument of periapsis for @ orbit (DEG)
*   AXIS - semi-major axis for @ orbit (KM)
*   ECC - eccentricity for @ orbit
*   INC - inclination for @ orbit (DEG)
*   MA - mean anomaly for @ orbit (DEG)
*   NODE - longitude of the ascending node for @ orbit (DEG)
*
* OTHER VARIABLES:
*   AKM - value of the semi-major axis in KM
*   CHOICE - reflects the users choice for operation
*   COUNT - DO loop counter
*   D2RAD - converts DEG to RAD
*   DU2KM - converts DUs to KM
*   ELIP2 - square of the eccentricity of the earth
*   KIND - differentiates between a ground station and three
*         satellites (0) and three satellites (1)
*   N - mean motion of the satellites (DU/TU)
*   S2TU - converts SEC to TU
*   SIZE - step size (SEC)
*   STATE - file number
*   STEPS - number of steps
*   T - chronological time over time span
*   TWOPI - twice the value of pi
*   VIS - value of the visibility function
*

```

```

*****
SUBROUTINE SATS(CHOICE,ELIP2,S2TU,SIZE,STEPS)

```

```

REAL*8 AKM,ARGP,AXIS,D2RAD,DU2KM,ECC,ELIP2,INC,MA,NODE
REAL*8 S2TU,SIZE,T(0:500),TWOPI,VIS(0:500),N
INTEGER CHOICE,COUNT,KIND,STATE,STEPS

```

```

OPEN(UNIT=1,FILE='POS1.DAT',STATUS='NEW')
OPEN(UNIT=2,FILE='POS2.DAT',STATUS='NEW')
OPEN(UNIT=3,FILE='POS3.DAT',STATUS='NEW')
OPEN(UNIT=4,FILE='POS4.DAT',STATUS='NEW')
OPEN(UNIT=60,FILE='TRACKER.DAT',STATUS='NEW')

```

```

*****
*                               CONSTANTS                               *
*****

```

```

TWOPI = 2.000*DACOS(-1.000)
D2RAD = (TWOPI/360.000)
DU2KM = 6378.13700

```

```

*****
* The following prompts the user to enter the values of the
* orbital elements for the satellites of interest.
*****

```

```

DO 20 COUNT = 1,CHOICE+1
  ECC = 0.000
  AXIS = 0.000
  INC = 0.000
  MA = 0.000
  NODE = 0.000
  ARGP = 0.000
  N = 0.000

```

```

WRITE(25,510)COUNT

```

```

WRITE(25,520)ECC
12 WRITE(*,*)'Mean Motion (rev/day) ==> '
   READ(*,*,ERR=12)N
      N = N*.05867301522d0
      AXIS = (1.0D0/(N**2.0D0))**(.333333D0)
      AKM = AXIS*DU2KM
      WRITE(25,530)AKM
13 WRITE(*,*)'Inclination ==> '
   READ(*,*,ERR=13)INC
   WRITE(25,540)INC
14 WRITE(*,*)'Mean Anomaly (DEG) ==> '
   READ(*,*,ERR=14)MA
   WRITE(25,550)MA
15 WRITE(*,*)'Longitude of the Ascending Node (DEG) ==> '
   READ(*,*,ERR=15)NODE
   WRITE(25,560)NODE
16 WRITE(*,*)'Argument of Periapsis (DEG) ==> '
   READ(*,*,ERR=16)ARGP
   WRITE(25,570)ARGP
   WRITE(25,*)
   WRITE(*,*)
   WRITE(*,*)'Please stand-by. Calculations are in progress.'
   WRITE(*,*)

```

```

*****
* The following converts the values entered from KM to DU, or from *
* DEG to RAD. *
*****

```

```

INC = INC*D2RAD
MA = MA*D2RAD
NODE = NODE*D2RAD
ARGP = ARGP*D2RAD

```

```

CALL POS(COUNT,ELIP2,S2TU,STEPS,AXIS,ECC,INC,N,NODE,ARGP,MA)
CLOSE(UNIT = COUNT)

```

```

20 CONTINUE

```

```

IF (CHOICE.EQ.0) THEN
  GOTO 400
ENDIF

```

```

*****
* In the following if loop, the computer calls the subroutine *
* necessary to check all options of visibility depending on the *
* number of satellites in consideration. *
*****

```

```

* 2 SATELLITES *
*****

```

```

WRITE(*,*)'The computer is in the process of determining 'ise'
WRITE(*,*)'and set times. Please stand by.'

```

```

IF (CHOICE.EQ.1) THEN
  CALL TWOVIS(STEPS,SIZE)
  OPEN(UNIT=10,FILE='VIS1.DAT',STATUS='OLD')
  DO 30 COUNT = 1,STEPS+1
    READ(10,*)T(COUNT),VIS(COUNT)

```

```

30 CONTINUE
   CLOSE(UNIT=10)
   WRITE(25,*)
   WRITE(25,600)

```

```

STATE = 41
OPEN(UNIT=41,FILE='STAT1.DAT',STATUS='NEW')
CALL BLEND(STATE,STEPS,T,VIS)

```

\*\*\*\*\*

```
ELSE IF (CHOICE.EQ.2) THEN
  CALL THRVIS(STEPS,SIZE)
  OPEN(UNIT=10,FILE='VIS1.DAT',STATUS='OLD')
  DO 40 COUNT = 1,STEPS+1
    READ(10,*)T(COUNT),VIS(COUNT)
40  CONTINUE
  CLOSE(UNIT=10)
  WRITE(25,*)
  WRITE(25,600)

  STATE = 41
  OPEN(UNIT=41,FILE='STAT1.DAT',STATUS='NEW')
  CALL BLEND(STATE,STEPS,T,VIS)
  CLOSE(UNIT=41)

  OPEN(UNIT=11,FILE='VIS2.DAT',STATUS='OLD')
  DO 50 COUNT = 1,STEPS+1
    READ(11,*)T(COUNT),VIS(COUNT)
50  CONTINUE
  CLOSE(UNIT=11)
  WRITE(25,*)
  WRITE(25,630)

  STATE=42
  OPEN(UNIT=42,FILE='STAT2.DAT',STATUS='NEW')
  CALL BLEND(STATE,STEPS,T,VIS)
  CLOSE(UNIT=42)

  OPEN(UNIT=12,FILE='VIS3.DAT',STATUS='OLD')
  DO 60 COUNT = 1,STEPS+1
    READ(12,*)T(COUNT),VIS(COUNT)
60  CONTINUE
  CLOSE(UNIT=12)
  WRITE(25,*)
  WRITE(25,620)

  STATE=43
  OPEN(UNIT=43,FILE='STAT3.DAT',STATUS='NEW')
  CALL BLEND(STATE,STEPS,T,VIS)
  CLOSE(UNIT=43)
  CLOSE(UNIT=60)

  KIND = 1
  STATE=41
  WRITE(25,*)
  WRITE(25,660)
  WRITE(25,665)
  CALL CVC3(KIND,STATE)
```

\*\*\*\*\*

\* 4 SATELLITES \*

\*\*\*\*\*

```
ELSE IF (CHOICE.EQ.3) THEN
  CALL FOURVIS(STEPS,SIZE)
  OPEN(UNIT=10,FILE='VIS1.DAT',STATUS='OLD')
  DO 70 COUNT = 1,STEPS+1
    READ(10,*)T(COUNT),VIS(COUNT)
70  CONTINUE
  CLOSE(UNIT=10)
  WRITE(25,*)
  WRITE(25,600)

  STATE=41
```

```
OPEN(UNIT=11, FILE='VIS2.DAT', STATUS='OLD')
DO 80 COUNT = 1, STEPS+1
  READ(11, *) T(COUNT), VIS(COUNT)
80 CONTINUE
CLOSE(UNIT=11)
WRITE(25, *)
WRITE(25, 610)

STATE=42
OPEN(UNIT=42, FILE='STAT2.DAT', STATUS='NEW')
CALL BLEND(STATE, STEPS, T, VIS)
CLOSE(UNIT=42)

OPEN(UNIT=12, FILE='VIS3.DAT', STATUS='OLD')
DO 90 COUNT = 1, STEPS+1
  READ(12, *) T(COUNT), VIS(COUNT)
90 CONTINUE
CLOSE(UNIT=12)
WRITE(25, *)
WRITE(25, 620)

STATE=43
OPEN(UNIT=43, FILE='STAT3.DAT', STATUS='NEW')
CALL BLEND(STATE, STEPS, T, VIS)
CLOSE(UNIT=43)

OPEN(UNIT=13, FILE='VIS4.DAT', STATUS='OLD')
DO 100 COUNT = 1, STEPS+1
  READ(13, *) T(COUNT), VIS(COUNT)
100 CONTINUE
CLOSE(UNIT=13)
WRITE(25, *)
WRITE(25, 630)

STATE=44
OPEN(UNIT=44, FILE='STAT4.DAT', STATUS='NEW')
CALL BLEND(STATE, STEPS, T, VIS)
CLOSE(UNIT=44)

OPEN(UNIT=14, FILE='VIS5.DAT', STATUS='OLD')
DO 110 COUNT = 1, STEPS+1
  READ(14, *) T(COUNT), VIS(COUNT)
110 CONTINUE
CLOSE(UNIT=14)
WRITE(25, *)
WRITE(25, 640)

STATE=45
OPEN(UNIT=45, FILE='STAT5.DAT', STATUS='NEW')
CALL BLEND(STATE, STEPS, T, VIS)
CLOSE(UNIT=45)

OPEN(UNIT=15, FILE='VIS6.DAT', STATUS='OLD')
DO 120 COUNT = 1, STEPS+1
  READ(15, *) T(COUNT), VIS(COUNT)
120 CONTINUE
CLOSE(UNIT=15)
WRITE(25, *)
WRITE(25, 650)

STATE=46
OPEN(UNIT=46, FILE='STAT6.DAT', STATUS='NEW')
```

```
STATE=41  
WRITE(25,*)  
WRITE(25,670)  
WRITE(25,665)  
CALL CVC4(STATE)
```

```
ENDIF
```

```
CLOSE(UNIT=60,STATUS='DELETE')
```

```
400 CONTINUE
```

```
RETURN
```

```
500 FORMAT(' elements of satellite #',I1,':')  
510 FORMAT(' INITIAL ORBITAL ELEMENTS OF SATELLITE #',I1,':')  
520 FORMAT(3X,'Eccentricity      : ',F12.10)  
530 FORMAT(3X,'Semi-major Axis : ',F15.8,' KM')  
540 FORMAT(3X,'Inclination      : ',F15.8,' DEG')  
550 FORMAT(3X,'Mean Anomaly     : ',F15.8,' DEG')  
560 FORMAT(3X,'Longitude of the Ascending Node : ',F15.8,' DEG')  
570 FORMAT(3X,'Argument of Periapsis      : ',F15.8,' DEG')  
600 FORMAT('Rise/Set Times between Satellite #1 and Satellite #2:')  
610 FORMAT('Rise/Set Times between Satellite #1 and Satellite #3:')  
620 FORMAT('Rise/Set Times between Satellite #1 and Satellite #4:')  
630 FORMAT('Rise/Set Times between Satellite #2 and Satellite #3:')  
640 FORMAT('Rise/Set Times between Satellite #2 and Satellite #4:')  
650 FORMAT('Rise/Set Times between Satellite #3 and Satellite #4:')  
660 FORMAT('COMMUNICATIONS STATUS BETWEEN SATELLITE #1, #2, AND #3:')  
665 FORMAT('(NOTE: NO ENTRIES OR MESSAGE BELOW INDICATES CONTINUOUS  
+ COMMUNICATIONS.))')  
670 FORMAT('COMMUNICATIONS STATUS BETWEEN SATELLITE #1, #2, #3, AND  
+ #4:')  
END
```

```
*
```

SUBROUTINE POS

PURPOSE: The purpose of this subroutine is to determine the position of a satellite given the classical orbital elements.

VARIABLES:

INPUT:

- ARGP - argument of periapsis for @ orbit (RAD)
- AXIS - semi-major axis for @ orbit (DU)
- COUNT - DO loop counter; indicates the satellite number
- ECC - eccentricity for @ orbit
- ELIP2 - square of the eccentricity of the earth
- INC - inclination for @ orbit (RAD)
- MA - mean anomaly for @ orbit (RAD)
- NODE - longitude of the ascending node for @ orbit (RAD)
- S2TU - time step (TU)
- STEPS - number of steps in the requested interval

LOCAL VARIABLES:

- COSARGP - cosine of the argument of periapsis
- COSFINALE - final value of the cosine of the eccentric anomaly
- COSINC - cosine of the inclination
- COSNODE - cosine of the longitude of the ascending node
- COSNU - cosine of the true anomaly
- DENOM - variable for the denominator of a given quotient
- DIF - difference between two value of the eccentric anomaly during Newton-Rhapson Iteration
- ECOSFE - eccentricity multiplied by the cosine of the finale value of the eccentric anomaly
- EN - first value of the eccentric anomaly for use in the Newton-Rhapson Iteration
- ENPLUSONE - second value of the eccentric anomaly as calculated in the Newton-Rhapson Iteration
- FINALE - final value of the eccentric anomaly reached by convergence of the Newton-Rhapson Iteration
- J2 -
- M - value of the mean anomaly at each time step
- MN - value of the mean anomaly calculated using a given value for the eccentric anomaly (used in the Newton-Rhapson Iteration)
- MULT - matrix used to rotate the position vectors from the perifocal reference frame to the geocentric reference frame
- N - mean motion of the satellite
- NANOM - anomalistic mean motion
- NEWARGP - propogated argument of periapsis
- NEWNODE - propogated longitude of the ascending node
- P - calculated value of the semi-latus rectum for the orbit
- R - calculated position vectors of the satellite
- REP - DO loop counter
- RPQW - position vectors of the satellite in the perifocal frame
- SINARGP - sine of the argument of periapsis
- SINFINE - sine of the final value of the eccentric anomaly
- SININC - sine of the inclination of the orbit
- SINNODE - sine of the longitude of the ascending node
- SINNU - sine of the true anomaly
- TWOPI - constant; two multiplied by pi

REFERENCES: Bates, Mueller, and White, 187, 219-222

\*\*\*\*\*  
SUBROUTINE POS(COUNT,ELIP2,S2TU,STEPS,AXIS,ECC,INC,N,NODE,ARGP,MA)

```

REAL*8 NODEDOT, NODE, NU, P, R(4), RPQW(4), S2TU, SINARGP
REAL*8 SINFINE, SININC, SINNODE, SINNU, TWOPI
INTEGER COUNT, REP, STEPS

```

```

*****
* The following computes values of variables which will be used *
* multiple times within the program. This is done to decrease the *
* amount of calculation necessary within the subroutine. *
*****

```

```

COSINC = DCOS(INC)
SININC = DSIN(INC)

J2      = .00108262D0
TWOPI   = 2.0D0*DACOS(-1.0D0)
P       = AXIS*(1.0D0 - (ECC**2.0D0))
N       = N*S2TU

```

```

*****
* The following calculates the values necessary to propagate the *
* satellite forward in its orbit. *
*****

```

```

NANOM   = N*(1.0D0+((1.5D0*J2*(DSQRT(1.0D0-(ECC**2.0D0)))/
+ (P**2.0D0))*(1.0D0-(1.5D0*(SININC**2.0D0))))))
NODEDOT = -(1.5D0*(J2/P**2.0D0)*COSINC)*NANOM
ARGPDOT = (1.5D0*(J2/P**2.0D0)*(2.0D0-(2.5D0*
+ (SININC**2.0D0)))*NANOM

```

```

DO 50 REP = 0, STEPS
  NEWNODE = DMOD((NODE + (NODEDOT)*REP), TWOPI)
  NEWARGP = DMOD((ARGP + (ARGPDOT)*REP), TWOPI)
  COSNODE = DCOS(NEWNODE)
  SINNODE = DSIN(NEWNODE)
  COSARGP = DCOS(NEWARGP)
  SINARGP = DSIN(NEWARGP)

```

```

*****
* The following 3x3 matrix is used to transform the position vector*
* from the PQW (perifocal) frame to the IJK (geocentric) frame. *
*****

```

```

MULT(1,1) = COSNODE*COSARGP - SINNODE*SINARGP*COSINC
MULT(1,2) = -COSNODE*SINARGP - SINNODE*COSARGP*COSINC
MULT(1,3) = SINNODE*SININC
MULT(2,1) = SINNODE*COSARGP + COSNODE*SINARGP*COSINC
MULT(2,2) = -SINNODF*SINARGP + COSNODE*COSARGP*COSINC
MULT(2,3) = -COSNODE*SININC
MULT(3,1) = SINARGP*SININC
MULT(3,2) = COSARGP*SININC
MULT(3,3) = COSINC

```

```

*****
* In the next loop, the Newton-Rhapson iteration is carried out in *
* order to determine the value of the eccentric anomaly at every *
* time step in the orbit of the satellite. *
*****

```

```

M = DMOD((MA + (NANOM*REP)), TWOPI)
EN = M
10 MN = EN - ECC*(DSIN(EN))
ENPLUSONE = (EN + ((M - MN)/(1.0D0 - (ECC*(DCOS(EN))))))
DIF = DABS(EN - ENPLUSONE)

```

```

IF (DIF.GT.(.000001D0)) THEN
  EN = ENPLUSONE
  GOTO 10
ELSE
  FINALE = ENPLUSONE
ENDIF

```

```

ECOSFE = ECC*COSFINE
DENOM  = AXIS*(1.000 - (ECC*COSFINE))

COSNU  = (ECC - COSFINE)/(ECOSFE - 1.000)
SINNU  = ((AXIS*(DSQRT(1.000 - ECC**2.000)))*SINFINE)/DENOM
NU     = DATAN2(SINNU,COSNU)

COSNU  = DCOS(NU)
SINNU  = DSIN(NU)
*****
* In the next three lines, the position vector in the perifocal *
* coordinate system is calculated. *
*****
RPQW(1) = (P*COSNU)/(1.000 + (ECC*COSNU))
RPQW(2) = (P*SINNU)/(1.000 + (ECC*COSNU))
RPQW(3) = 0.000
*****
* The next lines transform the RPQW vector to the geocentric *
* coordinate frame. *
*****
R(1) = RPQW(1)*MULT(1,1) + RPQW(2)*MULT(1,2)
R(2) = RPQW(1)*MULT(2,1) + RPQW(2)*MULT(2,2)
R(3) = (RPQW(1)*MULT(3,1) + RPQW(2)*MULT(3,2))*(1.000/DSQRT(1.000
+
- ELIP2))
R(4) = DSQRT((R(1)**2.000) + (R(2)**2.000) + (R(3)**2.000))

WRITE(COUNT,100)R(1),R(2),R(3),R(4)

50 CONTINUE
RETURN

100 FORMAT(F12.5,5X,F12.5,5X,F12.5,5X,F12.5)

END
*

```



```

-          SUBROUTINE TWOVIS          *
*
* PURPOSE: This subroutine examines the visibility opportunities *
*          between two satellites.   *
*
* VARIABLES: *
* INPUT: *
*   STEPS   - number of steps in the requested interval *
*   SIZE    - step size (SEC) *
*
* LOCAL VARIABLES: *
*   COUNT   - DO loop counter *
*   FILNUM  - file number *
*   PHI     - angle between position vectors R1 and R2 *
*   R1      - position vector to satellite #1 *
*   R2      - position vector to satellite #2 *
*   VIS     - visibility function between satellites #1 and #2 *
*
* REFERENCES: Larsen, 32 *
*            Alfano, Negrón, and Moore, 2 *
*            (These references apply to THRVIS and FOURVIS also) *
*****
SUBROUTINE TWOVIS(STEPS,SIZE)

INTEGER COUNT,STEPS
REAL*8 PHI,R1(4),R2(4),SIZE,VIS

OPEN(UNIT=1,FILE='POS1.DAT',STATUS='OLD')
OPEN(UNIT=2,FILE='POS2.DAT',STATUS='OLD')
*****
* VIS.DAT CONTAINS VISIBILITY DATA BETWEEN SATELLITES ONE AND TWO *
*****
OPEN(UNIT=10,FILE='VIS1.DAT',STATUS='NEW')

DO 20 COUNT = 0,STEPS
  READ(1,*)R1(1),R1(2),R1(3),R1(4)
  READ(2,*)R2(1),R2(2),R2(3),R2(4)
*****
* The next lines define the visibility function. According to this *
* idea, consider a plane tangent to the earth as the line below which*
* satellite to satellite visibility is not possible. The visibility *
* function calculated here compares angles to determine whether the *
* satellites are above or below the tangent plane. In this instance *
* only two satellites are involved. *
*****
*           Satellites #1 and #2 *
*****
  PHI = ((R1(1)*R2(1))+(R1(2)*R2(2))+(R1(3)*R2(3)))/(R1(4)*R2(4))

  IF (PHI.GT.(1.0D0)) THEN
    PHI = 1.0D0
  ELSE IF (PHI.LT.(-1.0D0)) THEN
    PHI = -1.0D0
  ENDIF

  VIS = (DACOS(1.0D0/R1(4)))+(DACOS(1.0D0/R2(4)))-(DACOS(PHI))
  WRITE(10,110)COUNT*SIZE,VIS

20 CONTINUE

CLOSE(UNIT=1)
CLOSE(UNIT=2)
CLOSE(UNIT=10)

```

110 FORMAT(I10,5X,F12.5)

END

\*

```

*          SUBROUTINE THRVIS          *
*
*  PURPOSE:  This subroutine examines the visibility opportunities
*            between three satellites.
*
*  VARIABLES:
*  INPUT:
*    STEPS   - number of steps in the requested interval
*    SIZE    - step size (SEC)
*
*  LOCAL VARIABLES:
*    COUNT   - DO loop counter
*    PHI     - angle between position vectors to two satellites
*    R1      - position vector to satellite #1
*    R2      - position vector to satellite #2
*    R3      - position vector to satellite #3
*    VIS     - visibility function between two satellites
*****
SUBROUTINE THRVIS(STEPS,SIZE)

  INTEGER COUNT,STEPS
  REAL*8  PHI,R1(4),R2(4),R3(4),SIZE,VIS

  OPEN(UNIT=1,FILE='POS1.DAT',STATUS='OLD')
  OPEN(UNIT=2,FILE='POS2.DAT',STATUS='OLD')
  OPEN(UNIT=3,FILE='POS3.DAT',STATUS='OLD')
*****
*  The following table lists filenames and their content
*
*  UNIT = 10; VIS1.DAT - visibility between 1 and 2
*  UNIT = 11; VIS2.DAT - visibility between 1 and 3
*  UNIT = 12; VIS3.DAT - visibility between 2 and 3
*****
  OPEN(UNIT=10,FILE='VIS1.DAT',STATUS='NEW')
  OPEN(UNIT=11,FILE='VIS2.DAT',STATUS='NEW')
  OPEN(UNIT=12,FILE='VIS3.DAT',STATUS='NEW')

  DO 20 COUNT = 0,STEPS
    READ(1,*)R1(1),R1(2),R1(3),R1(4)
    READ(2,*)R2(1),R2(2),R2(3),R2(4)
    READ(3,*)R3(1),R3(2),R3(3),R3(4)
*****
*  The following lines calculate the visibility function as explained *
*  in the subroutine TWOVIS.  Since three satellites are involved in *
*  this case, visibility checks must be made between satellites 1 and *
*  2, 2 and 3, and 1 and 3.
*****
*          Satellites #1 and #2
*****
  PHI = ((R1(1)*R2(1))+(R1(2)*R2(2))+(R1(3)*R2(3)))/(R1(4)*R2(4))

  IF (PHI.GT.(1.0D0)) THEN
    PHI = 1.0D0
  ELSE IF (PHI.LT.(-1.0D0)) THEN
    PHI = -1.0D0
  ENDIF

  VIS = (DACOS(1.0D0/R1(4))) + (DACOS(1.0D0/R2(4))) - (DACOS(PHI))
  WRITE(10,110)COUNT*SIZE,VIS
*****
*          Satellites #1 and #3
*****
  PHI = ((R1(1)*R3(1))+(R1(2)*R3(2))+(R1(3)*R3(3)))/(R1(4)*R3(4))

```

```
ELSE IF (PHI.LT.(-1.000)) THEN
  PHI = -1.000
ENDIF
```

```
VIS = (DACOS(1.000/R1(1))) + (DACOS(1.000/R3(1))) - (DACOS(PHI))
WRITE(11,110)COUNT*SIZE,VIS
```

```
*****
*                               *
*           Satellites #2 and #3           *
*****
```

```
PHI = ((R2(1)*R3(1))+(R2(2)*R3(2))+(R2(3)*R3(3)))/(R2(4)*R3(4))
```

```
IF (PHI.GT.(1.000)) THEN
  PHI = 1.000
ELSE IF (PHI.LT.(-1.000)) THEN
  PHI = -1.000
ENDIF
```

```
VIS = (DACOS(1.000/R2(4)))+(DACOS(1.000/R3(4)))-(DACOS(PHI))
WRITE(12,110)COUNT*SIZE,VIS
```

```
20 CONTINUE
```

```
CLOSE(UNIT = 1)
CLOSE(UNIT = 2)
CLOSE(UNIT = 3)
CLOSE(UNIT = 10)
CLOSE(UNIT = 11)
CLOSE(UNIT = 12)
```

```
RETURN
```

```
110 FORMAT(110,3X,F12.5)
```

```
END
```

```
*
```

```

*          SUBROUTINE FOURVIS          *
*
* PURPOSE: This subroutine examines the visibility opportunities *
*          between four satellites.   *
*
* VARIABLES:
* INPUT:
*   STEPS   - number of steps in the requested interval *
*   SIZE    - step size (SEC) *
*
* LOCAL VARIABLES:
*   COUNT   - DO loop counter *
*   PHI     - angle between position vectors to two satellites *
*   R1      - position vector to satellite #1 *
*   R2      - position vector to satellite #2 *
*   R3      - position vector to satellite #3 *
*   R4      - position vector to satellite #4 *
*   VIS     - visibility function between two satellites *

```

```

*****
SUBROUTINE FOURVIS(STEPS,SIZE)

```

```

INTEGER COUNT,STEPS
REAL*8 PHI,R1(4),R2(4),R3(4),R4(4),SIZE,VIS

```

```

OPEN(UNIT=1,FILE='POS1.DAT',STATUS='OLD')
OPEN(UNIT=2,FILE='POS2.DAT',STATUS='OLD')
OPEN(UNIT=3,FILE='POS3.DAT',STATUS='OLD')
OPEN(UNIT=4,FILE='POS4.DAT',STATUS='OLD')

```

```

*****

```

```

* The following table lists filenames and their content *
* UNIT = 10; VIS1.DAT - visibility between 1 and 2 *
* UNIT = 11; VIS2.DAT - visibility between 1 and 3 *
* UNIT = 12; VIS3.DAT - visibility between 1 and 4 *
* UNIT = 13; VIS4.DAT - visibility between 2 and 3 *
* UNIT = 14; VIS5.DAT - visibility between 2 and 4 *
* UNIT = 15; VIS6.DAT - visibility between 3 and 4 *

```

```

*****

```

```

OPEN(UNIT=10,FILE='VIS1.DAT',STATUS='NEW')
OPEN(UNIT=11,FILE='VIS2.DAT',STATUS='NEW')
OPEN(UNIT=12,FILE='VIS3.DAT',STATUS='NEW')
OPEN(UNIT=13,FILE='VIS4.DAT',STATUS='NEW')
OPEN(UNIT=14,FILE='VIS5.DAT',STATUS='NEW')
OPEN(UNIT=15,FILE='VIS6.DAT',STATUS='NEW')

```

```

DO 20 COUNT = 0,STEPS
  READ(1,*)R1(1),R1(2),R1(3),R1(4)
  READ(2,*)R2(1),R2(2),R2(3),R2(4)
  READ(3,*)R3(1),R3(2),R3(3),R3(4)
  READ(4,*)R4(1),R4(2),R4(3),R4(4)

```

```

*****

```

```

* The following lines calculate the visibility function as explained *
* in the subroutine TNOVIS. Since four satellites are involved in *
* this case, visibility checks must be made between satellites 1 and *
* 2, 1 and 3, 1 and 4, 2 and 3, 2 and 4, and 3 and 4. *

```

```

*****

```

```

*           Satellites #1 and #2 *

```

```

*****

```

```

PHI = ((R1(1)*R2(1))+(R1(2)*R2(2))+(R1(3)*R2(3)))/(R1(4)*R2(4))

```

```

IF (PHI.GT.(1.0D0)) THEN
  PHI = 1.0D0
ELSE IF (PHI.LT.(-1.0D0)) THEN
  PHI = -1.0D0

```

```

*****
*                               Satellites #1 and #3                               *
*****
PHI = ((R1(1)*R3(1))+(R1(2)*R3(2))+(R1(3)*R3(3)))/(R1(4)*R3(4))

IF (PHI.GT.(1.000)) THEN
  PHI = 1.000
ELSE IF (PHI.LT.(-1.000)) THEN
  PHI = -1.000
ENDIF

VIS = (DACOS(1.000/R1(4))) + (DACOS(1.000/R3(4))) - (DACOS(PHI))
WRITE(11,110)COUNT*SIZE,VIS
*****
*                               Satellites #1 and #4                               *
*****
PHI = ((R1(1)*R4(1))+(R1(2)*R4(2))+(R1(3)*R4(3)))/(R1(4)*R4(4))

IF (PHI.GT.(1.000)) THEN
  PHI = 1.000
ELSE IF (PHI.LT.(-1.000)) THEN
  PHI = -1.000
ENDIF

VIS = (DACOS(1.000/R1(4))) + (DACOS(1.000/R4(4))) - (DACOS(PHI))
WRITE(12,110)COUNT*SIZE,VIS
*****
*                               Satellites #2 and #3                               *
*****
PHI = ((R2(1)*R3(1))+(R2(2)*R3(2))+(R2(3)*R3(3)))/(R2(4)*R3(4))

IF (PHI.GT.(1.000)) THEN
  PHI = 1.000
ELSE IF (PHI.LT.(-1.000)) THEN
  PHI = -1.000
ENDIF

VIS = (DACOS(1.000/R2(4))) + (DACOS(1.000/R3(4))) - (DACOS(PHI))
WRITE(13,110)COUNT*SIZE,VIS
*****
*                               Satellites #2 and #4                               *
*****
PHI = ((R2(1)*R4(1))+(R2(2)*R4(2))+(R2(3)*R4(3)))/(R2(4)*R4(4))

IF (PHI.GT.(1.000)) THEN
  PHI = 1.000
ELSE IF (PHI.LT.(-1.000)) THEN
  PHI = -1.000
ENDIF

VIS = (DACOS(1.000/R2(4))) + (DACOS(1.000/R4(4))) - (DACOS(PHI))
WRITE(14,110)COUNT*SIZE,VIS
*****
*                               Satellites #3 and #4                               *
*****
PHI = ((R3(1)*R4(1))+(R3(2)*R4(2))+(R3(3)*R4(3)))/(R3(4)*R4(4))

IF (PHI.GT.(1.000)) THEN
  PHI = 1.000
ELSE IF (PHI.LT.(-1.000)) THEN
  PHI = -1.000
ENDIF

```

```
WRITE(15,110)COUNT*SIZE,VIS
```

```
20 CONTINUE
```

```
CLOSE(UNIT = 1)  
CLOSE(UNIT = 2)  
CLOSE(UNIT = 3)  
CLOSE(UNIT = 4)  
CLOSE(UNIT = 10)  
CLOSE(UNIT = 11)  
CLOSE(UNIT = 12)  
CLOSE(UNIT = 13)  
CLOSE(UNIT = 14)  
CLOSE(UNIT = 15)
```

```
RETURN
```

```
110 FORMAT(I10,5X,F12.5)
```

```
END
```

SUBROUTINE BLEND

PURPOSE: The purpose of this subroutine is to parabolically blend the points of the visibility function. The equation of the blended curve will then be solved for real roots. If a real root exists, it represents either a rise or set time for the objects of interest.

VARIABLES:

INPUT:

- STATE - specifies a file unit number
- STEPS - number of steps
- T - chronological time over time span
- VIS - value of the visibility function

LOCAL VARIABLES:

- A,B - variables used in the solution of the closed form cubic equation (Escobal, 432-434)
- ALPHA - coefficients of the cubic
- CHECK - slightly later time than cross. The value of check is used to determine if a horizon crossing is a rise or set.
- COSPHI - cosine of the angle phi used in solution of the cubic equation
- COUNT - do loop counter
- CROSS - time of a rise or set
- D120 - 120 DEG changed to radians
- D240 - 240 DEG changed to radians
- DELTA - see closed form solution of cubic equation
- EVENT - 1 if the crossing is a rise; 0 if the crossing is a set
- EO - see closed form solution of cubic equation
- INC - increment of time T
- O - coefficient of cubed variable in cubic equation
- P - coefficient of squared variable in cubic equation
- PHI - angle used in calculation of roots of cubic equation
- PI - mathematical constant pi
- Q - coefficient of the first degree variable in the cubic equation
- QTNT - portion of the quadratic equation under the radical
- R - constant value in the cubic equation
- ROOT1,ROOT2,ROOT3 - possible roots of the cubic equation
- SINPHI - sine of the angle phi used in solution of the cubic equation
- Z1,Z2,Z3 - variables used in determining roots of the cubic equation

REFERENCES: Adams and Rogers, 278-284  
 Alfano, Negrón, and Moore, 4-5  
 Escobal, 431-433

SUBROUTINE BLEND(STATE,STEPS,T,VIS)

INTEGER COUNT,STATE,STEPS

REAL\*8 A,ARG1,ARG2,B,ALP2(0:3),ALPHA(0:3),CHECK,COSPHI,CROSS

REAL\*8 D120,D240,DELTA,EO,EVENT,INC,P,PHI,PI,Q,QTNT,VALA,VALB

REAL\*8 R,ROOT1,ROOT2,ROOT3,SINPHI,T(0:STEPS+2)

REAL\*8 VIS(0:STEPS+2),Z1,Z2,Z3

CONSTANTS

PI = DACOS(-1.0D0)  
 D120 = 120.0D0\*(PI/180.0D0)



```

* The following lines assign values to the first and last points in *
* in the visibility curve. These points fall immediately before and *
* immediately after the endpoints of interest. This insures no *
* portion of the curve will be omitted when testing for horizon *
* crossings.

```

```

*****
T(0) = T(1)
VIS(0) = VIS(1)
T(STEPS-2) = T(STEPS+1)
VIS(STEPS-2) = VIS(STEPS+1)
*****

```

```

* In the following do loop, the coefficients of the cubic equation *
* are determined using parabolic blending. *
*****

```

```

DO 10 COUNT = 1, STEPS
  ALPHA(3) = (-.50D0)*VIS(COUNT-1)+(1.50D0)*VIS(COUNT)+(-1.50D0)*
+           VIS(COUNT+1)+(.50D0)*VIS(COUNT+2)
  ALPHA(2) = VIS(COUNT-1)+(-2.50D0)*VIS(COUNT)+(2.0D0)*
+           VIS(COUNT+1)+(-.50D0)*VIS(COUNT+2)
  ALPHA(1) = (-.50D0)*VIS(COUNT-1)+(.50D0)*VIS(COUNT+1)
  ALPHA(0) = VIS(COUNT)
*****

```

```

* In the event that the equation is a quadratic equation instead of *
* a cubic equation, the following lines determine the roots of the *
* quadratic.
*****

```

```

IF (DABS(ALPHA(3)).LE.(0.000001D0).AND.DABS(ALPHA(2)).LE.
+ (0.000001D0).AND.DABS(ALPHA(1)).LT.(.000001D0)) THEN
  ROOT1 = -1.0D0
  ROOT2 = -1.0D0
  ROOT3 = -1.0D0
  GOTO 5
ELSE IF (DABS(ALPHA(3)).LE.(0.000001D0).AND.DABS(ALPHA(2)).LE.
+ (0.000001D0)) THEN
  ROOT1 = -ALPHA(0)/ALPHA(1)
  ROOT2 = -1.0D0
  ROOT3 = -1.0D0
  GOTO 5
ELSE IF (DABS(ALPHA(3)).LE.(0.000001D0)) THEN
  QINT = (ALPHA(1)**2.0D0)-(1.0D0*ALPHA(2)*ALPHA(0))
  IF (QINT.GE.(0.0D0)) THEN
    ROOT1 = (-ALPHA(1)+DSQRT(QINT))/(2.0D0*ALPHA(2))
    ROOT2 = (-ALPHA(1)-DSQRT(QINT))/(2.0D0*ALPHA(2))
    ROOT3 = -1.0D0
    GOTO 5
  ELSE
    ROOT1 = -1.0D0
    ROOT2 = -1.0D0
    ROOT3 = -1.0D0
    GOTO 5
  ENDIF
ENDIF
*****

```

```

* The following is a closed form solution to the cubic equation.. *
*****

```

```

P = ALPHA(2)/ALPHA(3)
Q = ALPHA(1)/ALPHA(3)
R = ALPHA(0)/ALPHA(3)
A = (1.0D0/3.0D0)*((3.0D0*Q) - (P**2.0D0))
B = (1.0D0/27.0D0)*((2.0D0*(P**3.0D0))- (9.0D0*P*Q)+ (27.0D0*R))
DELTA = ((A**3.0D0)/27.0D0) + ((B**2.0D0)/1.0D0)

```

```

IF (DELTA.GT.(0.0D0)) THEN

```

```

IF (ARG1.LT.(0.0D0)) THEN
  VALA = -((DABS(ARG1))**.33332D0)
ELSE
  VALA = ARG1**.33333D0
ENDIF

IF (ARG2.LT.(0.0D0)) THEN
  VALB = -((DABS(ARG2))**.33333D0)
ELSE
  VALB = ARG2**.33333D0
ENDIF

Z1 = VALA + VALB
ROOT1 = Z1 - (P/3.0D0)
ROOT2 = -1.0D0
ROOT3 = -1.0D0
ELSE IF (DELTA.EQ.(0.0D0)) THEN
  ARG1 = -B/2.0D0
  ARG2 = B/2.0D0

  IF (ARG1.LT.(0.0D0)) THEN
    Z1 = -2.0D0*((DABS(ARG1))**.33333D0)
  ELSE IF (ARG1.GT.(0.0D0)) THEN
    Z1 = 2.0D0*((DABS(ARG1))**.33333D0)
  ENDIF

  IF (ARG2.LT.(0.0D0)) THEN
    Z2 = -((DABS(ARG2))**.33333D0)
  ELSE IF (ARG2.GT.(0.0D0)) THEN
    Z2 = ARG2**.33333D0
  ENDIF

  ROOT1 = Z1 - (P/3.0D0)
  ROOT2 = Z2 - (P/3.0D0)
  ROOT3 = -1.0D0
ELSE IF (DELTA.LT.(0.0D0)) THEN
  ARG1 = -(A**3.0D0)/27.0D0
  ARG2 = -(A/3.0D0)

  IF (ARG1.LT.(0.0D0).OR.ARG2.LT.(0.0D0)) THEN
    ROOT1 = -1.0D0
    ROOT2 = -1.0D0
    ROOT3 = -1.0D0
  ELSE
    EO = 2.0D0*DSQRT(ARG2)
    COSPHI = -B/(2.0D0*DSQRT(ARG1))
    SINPHI = DSQRT(1.0D0-(COSPHI**2.0D0))
    PHI = DATAN2(SINPHI,COSPHI)
    Z1 = EO*DCOS(PHI/3.0D0)
    Z2 = EO*DCOS((PHI/3.0D0)+D120)
    Z3 = EO*DCOS((PHI/3.0D0)+D240)

    ROOT1 = Z1 - (P/3.0D0)
    ROOT2 = Z2 - (P/3.0D0)
    ROOT3 = Z3 - (P/3.0D0)
  ENDIF
ENDIF

```

5 CONTINUE

```

*****
* If a positive root is found, parabolic blending is used to deter- *
* a cubic equation based on the times corresponding to the visibility*
* values of interest. The value of this cubic equation evaluated at *
* the root of the equation above (ROOT1,ROOT2, or ROOT3 above) to *
* find the exact time of crossing. The same equation is evaluated *

```

```

*****
IF (ROOT1.GE.0.AND.ROOT1.LE.1) THEN
ALP2(3) = (-.5000)*T(COUNT-1)+(1.5000)*T(COUNT)+(-1.5000)*
+ T(COUNT+1)+(.5000)*T(COUNT+2)
ALP2(2) = T(COUNT-1)+(-2.5000)*T(COUNT)+(2.0000)*
+ T(COUNT+1)+(-.5000)*T(COUNT+2)
ALP2(1) = (-.5000)*T(COUNT-1)+(.5000)*T(COUNT+1)
ALP2(0) = T(COUNT)

CROSS = ((ROOT1**3.000)*ALP2(3))+((ROOT1**2.000)*ALP2(2))+
+ (ROOT1*ALP2(1))+ALP2(0)
ROOT1 = ROOT1+INC
CHECK = ((ROOT1**3.000)*ALPHA(3))+((ROOT1**2.000)*ALPHA(2))+
+ (ROOT1*ALPHA(1))+ALPHA(0)
IF (CROSS.GE.(T(COUNT)).AND.CROSS.LE.(T(COUNT+1))) THEN
IF (CHECK.GT.(0.000)) THEN
WRITE(25,100)CROSS
EVENT = 1.000
WRITE(60,*)CROSS,EVENT
WRITE(STATE,*)CROSS,EVENT
ELSE
WRITE(25,110)CROSS
EVENT = 0.000
WRITE(60,*)CROSS,EVENT
WRITE(STATE,*)CROSS,EVENT
ENDIF
ENDIF
ENDIF

IF (ROOT2.GE.0.AND.ROOT2.LE.1) THEN
ALP2(3) = (-.5000)*T(COUNT-1)+(1.5000)*T(COUNT)+(-1.5000)*
+ T(COUNT+1)+(.5000)*T(COUNT+2)
ALP2(2) = T(COUNT-1)+(-2.5000)*T(COUNT)+(2.0000)*
+ T(COUNT+1)+(-.5000)*T(COUNT+2)
ALP2(1) = (-.5000)*T(COUNT-1)+(.5000)*T(COUNT+1)
ALP2(0) = T(COUNT)

CROSS = ((ROOT2**3)*ALP2(3))+((ROOT2**2)*ALP2(2))+((ROOT2*
+ ALP2(1))+ALP2(0)
ROOT2 = ROOT2+INC
CHECK = ((ROOT2**3)*ALPHA(3))+((ROOT2**2)*ALPHA(2))+((ROOT2*
+ ALPHA(1))+ALPHA(0)
IF (CROSS.GE.(T(COUNT)).AND.CROSS.LE.(T(COUNT+1))) THEN
IF (CHECK.GT.(0.000)) THEN
WRITE(25,100)CROSS
EVENT = 1.000
WRITE(60,*)CROSS,EVENT
WRITE(STATE,*)CROSS,EVENT
ELSE
WRITE(25,110)CROSS
EVENT = 0.000
WRITE(60,*)CROSS,EVENT
WRITE(STATE,*)CROSS,EVENT
ENDIF
ENDIF
ENDIF

IF (ROOT3.GE.0.AND.ROOT3.LE.1) THEN
ALP2(3) = (-.5000)*T(COUNT-1)+(1.5000)*T(COUNT)+(-1.5000)*
+ T(COUNT+1)+(.5000)*T(COUNT+2)
ALP2(2) = T(COUNT-1)+(-2.5000)*T(COUNT)+(2.0000)*
+ T(COUNT+1)+(-.5000)*T(COUNT+2)
ALP2(1) = (-.5000)*T(COUNT-1)+(.5000)*T(COUNT+1)

```

```

+      ALP2(1))+ALP2(0)
ROOT3 = ROOT3+INC
CHECK = ((ROOT3**3)*ALPHA(3))+((ROOT3**2)*ALPHA(2))+((ROOT3*
+      ALPHA(1))+ALPHA(0)
IF (CROSS.GE.(T(COUNT)).AND.CROSS.LE.(T(COUNT+1))) THEN
  IF (CHECK.GT.(0.0D0)) THEN
    WRITE(25,100)CROSS
    EVENT = 1.0D0
    WRITE(60,*)CROSS,EVENT
    WRITE(STATE,*)CROSS,EVENT
  ELSE
    WRITE(25,110)CROSS
    EVENT = 0.0D0
    WRITE(60,*)CROSS,EVENT
    WRITE(STATE,*)CROSS,EVENT
  ENDIF
ENDIF
ENDIF
ENDIF

10 CONTINUE

100 FORMAT(F12.5,' SEC - RISE')
110 FORMAT(F12.5,' SEC - SET')

RETURN
END

```

SUBROUTINE GSSVC2

```

*
* PURPOSE: The purpose of this subroutine is to accomplish a
*          check for constant visibility between a ground station
*          and 2 satellites.
*
* VARIABLES:
*   INPUT:
*     STATE - File number specifier
*
*   LOCAL VARIABLES:
*     CHANGE - specifies when communication is broken or re-
*             established
*     COUNT - do loop counter
*     E1,E2 - Specifies rise/set time for visibility function
*            a ground station and 2 satellites
*     EVENT - Array of all rise/set times
*     IN - do loop counter
*     RS1,RS2 - 1 if the event is a rise; 0 if the event is a set
*     SUM - Sum of RS1 and RS2 at a given event
*     T - time zero of any selected visibility function
*        - in this subroutine T is a place holder
*     TEMP - Temporary value used in sort loop
*     TOTAL - Total number of events
*     TOT1 - Number of events between Ground Station #? and
*           Satellite #1
*     TOT2 - Number of events between Ground Station #? and
*           Satellite #2
*     TT - Holds all event times and the corresponding
*          status of the visibility function for each
*          viewing path
*     V - the first value of the selected visibility
*        function
*     X - do loop counter

```

\*\*\*\*\*

SUBROUTINE GSSVC2(STATE)

```

REAL*8 E1(0:100),E2(0:100),EVENT(0:500)
REAL*8 RS1(0:100),RS2(0:100),SUM,T,TEMP,TT(0:500,2),V
INTEGER COUNT,CHANGE,CHECK,IN,STATE,TOTAL,TOT1,TOT2,X

```

```

OPEN(UNIT=41,FILE='STAT1.DAT',STATUS='OLD')
OPEN(UNIT=42,FILE='STAT2.DAT',STATUS='OLD')
OPEN(UNIT=60,FILE='TRACKER.DAT',STATUS='OLD')

```

REWIND(UNIT=60)

```

OPEN(UNIT=10,FILE='VIS1.DAT',STATUS='OLD')
OPEN(UNIT=11,FILE='VIS2.DAT',STATUS='OLD')

```

```

WRITE(*,*)'The computer is in the process of determining whether'
WRITE(*,*)'or not continuous communications is possible. When '
WRITE(*,*)'the program is finished running, results can be found'
WRITE(*,*)'in the file RISESET.DAT.'

```

\*\*\*\*\*

```

*          GROUND STATION AND SATELLITE #1
*          *****

```

```

TOT1 = 0
DO 20 COUNT = 1,100
  READ(STATE,*,END=25)E1(COUNT),RS1(COUNT)
  TOT1 = COUNT
20 CONTINUE

```

```

READ(10,*)T,V
IF (V.GT.(0.000)) THEN
  RS1(0) = 1.000
ELSE
  RS1(0) = 0.000
ENDIF
ELSE IF (RS1(1).EQ.(0.000)) THEN
  RS1(0) = 1.000
ELSE
  RS1(0) = 0.000
ENDIF
*****
*          GROUND STATION AND SATELLITE #2          *
*****
TOT2 = 0
DO 30 COUNT = 1,100
  READ(STATE+1,*,END=35)E2(COUNT),RS2(COUNT)
  TOT2 = COUNT
30 CONTINUE
35 E2(0) = 0.000

IF (TOT2.EQ.0) THEN
  READ(11,*)T,V
  IF (V.GT.(0.000)) THEN
    RS2(0) = 1.000
  ELSE
    RS2(0) = 0.000
  ENDIF
ELSE IF (RS2(1).EQ.(0.000)) THEN
  RS2(0) = 1.000
ELSE
  RS2(0) = 0.000
ENDIF
*****
* The following calculates the total number of events between the *
* ground station and two satellites.                               *
*****
TOTAL = TOT1+TOT2

CHECK = 0
DO 45 COUNT = 1,500
  READ(60,*,END=48)EVENT(COUNT)
  CHECK = COUNT
45 CONTINUE
48 CONTINUE

IF (CHECK.EQ.0) THEN
  WRITE(25,*)'There were no visibility opportunities with this
+ configuration.'
  GOTO 140
ENDIF
*****
* The following orders all the events from the smallest to the *
* largest.                                                         *
*****
DO 60 COUNT = 1,TOTAL-1
  DO 50 X = 1,TOTAL-1
    IF (EVENT(X+1).LT.EVENT(X)) THEN
      TEMP      = EVENT(X+1)
      EVENT(X+1) = EVENT(X)
      EVENT(X)   = TEMP
    ENDIF
  50 CONTINUE

```

```
*****
* The following determines the status of communications for each *
* possible visibility path at each event. *
*****
```

```
DO 80 COUNT = 0, TOTAL
  IF (TOT1.EQ.0) THEN
    TT(COUNT,1) = RS1(0)
    GOTO 80
  ENDIF
DO 70 IN = 0, TOT1 - 1
  IF (EVENT(COUNT).GE.E1(IN).AND.EVENT(COUNT).LT.E1(IN+1)) THEN
    TT(COUNT,1) = RS1(IN)
    GOTO 80
  ELSE IF (EVENT(COUNT).EQ.E1(IN+1)) THEN
    TT(COUNT,1) = RS1(IN+1)
    GOTO 80
  ELSE IF (EVENT(COUNT).GT.E1(IN+1).AND.IN.EQ.(TOT1-1)) THEN
    TT(COUNT,1) = RS1(IN+1)
    GOTO 80
  ENDIF
70 CONTINUE
80 CONTINUE
```

```
DO 120 COUNT = 0, TOTAL
  IF (TOT2.EQ.0) THEN
    TT(COUNT,2) = RS2(0)
    GOTO 120
  ENDIF
DO 110 IN = 0, TOT2 - 1
  IF (EVENT(COUNT).GE.E2(IN).AND.EVENT(COUNT).LT.E2(IN+1)) THEN
    TT(COUNT,2) = RS2(IN)
    GOTO 120
  ELSE IF (EVENT(COUNT).EQ.E2(IN+1)) THEN
    TT(COUNT,2) = RS2(IN+1)
    GOTO 120
  ELSE IF (EVENT(COUNT).GT.E2(IN+1).AND.IN.EQ.(TOT2-1)) THEN
    TT(COUNT,2) = RS2(IN+1)
    GOTO 120
  ENDIF
110 CONTINUE
120 CONTINUE
```

CHANGE = 1

```
*****
* The next DO loop determines if constant communication is *
* possible. If the sum is equal to zero, communication is not *
* possible. If the sum is greater than or equal to one, communi- *
* cation is possible. *
*****
```

```
DO 130 COUNT = 0, TOTAL
  SUM = TT(COUNT,1) + TT(COUNT,2)
  IF (SUM.EQ.(0.000).AND.CHANGE.EQ.1) THEN
    WRITE(25,*)'Communications broken at ',EVENT(COUNT)
    CHANGE = 0
  ENDIF

  IF (SUM.GE.(1.000).AND.CHANGE.EQ.0) THEN
    WRITE(25,*)'Communications re-established at ',EVENT(COUNT)
    CHANGE = 1
  ENDIF
130 CONTINUE
140 CONTINUE
```

CLOSE(UNIT=10,STATUS='DELETE')  
CLOSE(UNIT=11,STATUS='DELETE')

RETURN  
END



```

*****
*
*           SUBROUTINE CVC3
*
* PURPOSE:  The purpose of this subroutine is to accomplish a
*           check for constant visibility between 3 satellites.
*           For visibility to be maintained, at least two
*           paths of visibility must be maintained at all times.
*
* VARIABLES:
* INPUT:
*   KIND      - specifies whether 3 satellites are involved (1)
*              or a ground station and 3 satellites are
*              involved (0)
*   STATE     - File number specifier
*
* LOCAL VARIABLES:
*   CHANGE    - specifies when communication is broken or re-
*              established
*   COUNT     - do loop counter
*   E1,...,E3 - Specifies rise/set time for visibility function
*              between 3 satellites or a ground station and
*              3 satellites
*   EVENT     - Array of all rise/set times
*   IN        - do loop counter
*   RS1,...RS3 - 1 if the event is a rise; 0 if the event is a set
*   SUM       - Sum of RS1...RS3 at a given event
*   T         - time zero read from file of visibility data
*              - in this subroutine, T serves only as a place
*              holder
*   TEMP      - Temporary value used in sort loop
*   TOTAL     - Total number of events
*   TOT1      - Number of events between Satellite #1 and
*              Satellite #2 or Ground Station #? and
*              Satellite #1
*   TOT2      - Number of events between Satellite #2 and
*              Satellite #3 or Ground Station #? and
*              Satellite #2
*   TOT3      - Number of events between Satellite #1 and
*              Satellite #3 or Ground Station #? and
*              Satellite #3
*   TT        - Holds all event times and the corresponding
*              status of the visibility function for each
*              viewing path
*   V         - value of the chosen visibility function at time
*              equal to zero
*   X         - do loop counter
*****

```

```

SUBROUTINE CVC3(KIND,STATE)

```

```

REAL*8  E1(0:100),E2(0:100),E3(0:100),EVENT(0:500),RS1(0:100)
REAL*8  RS2(0:100),RS3(0:100),SUM,T,TEMP,TT(0:500,3),V
INTEGER CHANGE,CCUNT,IN,KIND,STATE,TOTAL,TOT1,TOT2,TOT3,X

```

```

IF (KIND.EQ.1) THEN
  WRITE(*,*)
  WRITE(*,*)'The computer is in the process of determining'
  WRITE(*,*)'whether or not constant communications is'
  WRITE(*,*)'maintained. When the program has finished'
  WRITE(*,*)'running, results may be found in RISESET.DAT.'

```

```

ENDIF

```

```

OPEN(UNIT=41,FILE='STAT1.DAT',STATUS='OLD')

```

```

OPEN(UNIT=60,FILE='TRACKER.DAT',STATUS='OLD')

IF (KIND.EQ.0) THEN
  REWIND(UNIT=60)
ENDIF

OPEN(UNIT=10,FILE='VIS1.DAT',STATUS='OLD')
OPEN(UNIT=11,FILE='VIS2.DAT',STATUS='OLD')
OPEN(UNIT=12,FILE='VIS3.DAT',STATUS='OLD')
*****
* The following DO loops read the rise and set times for each vis- *
* ibility path. *
*****
*           SATELLITE #1 AND SATELLITE #2 *
*           GROUND STATION AND SATELLITE #1 *
*****
  TOT1 = 0
  DO 20 COUNT = 1,100
    READ(STATE,*,END=25)E1(COUNT),RS1(COUNT)
    TOT1 = COUNT
  20 CONTINUE
  25 E1(0) = 0.000
*****
* If there were no rise or set times, the file containing the vis- *
* ibility data must be referenced to determine if the two objects *
* of interest are always in view or never in view. *
*****
  IF (TOT1.EQ.0) THEN
    READ(10,*)T,V
    IF (V.GT.(0.000)) THEN
      RS1(0)=1.000
    ELSE
      RS1(0)=0.000
    ENDIF
*****
* The following line checks the first condition of the first rise *
* or set time then assigns the appropriate condition to time zero. *
*****
    ELSE IF (RS1(1).EQ.(0.000)) THEN
      RS1(0) = 1.000
    ELSE
      RS1(0) = 0.000
    ENDIF
*****
*           SATELLITE #1 AND SATELLITE #3 *
*           GROUND STATION AND SATELLITE #2 *
*****
  TOT2 = 0
  DO 30 COUNT = 1,100
    READ(STATE+1,*,END=35)E2(COUNT),RS2(COUNT)
    TOT2 = COUNT
  30 CONTINUE
  35 E2(0) = 0.000

  IF (TOT2.EQ.0) THEN
    READ(11,*)T,V
    IF (V.GT.(0.000)) THEN
      RS2(0)=1.000
    ELSE
      RS2(0)=0.000
    ENDIF
*****
* The following line checks the first condition of the first rise *
* or set time then assigns the appropriate condition to time zero. *
*****

```

```

        RS2(0) = 1.000
    ELSE
        RS2(0) = 0.000
    ENDIF
*****
*           SATELLITE #2 AND SATELLITE #3           *
*           GROUND STATION AND SATELLITE #3         *
*****
    TOT3 = 0
    DO 40 COUNT = 1,100
        READ(STATE+2,*,END=15)E3(COUNT),RS3(COUNT)
        TOT3 = COUNT
    40 CONTINUE
    45 E3(0) = 0.000

    IF (TOT3.EQ.0) THEN
        READ(12,*)T,V
        IF (V.GT.(0.000)) THEN
            RS3(0)=1.000
        ELSE
            RS3(0)=0.000
        ENDIF
*****
* The following line checks the first condition of the first rise *
* or set time then assigns the appropriate condition to time zero. *
*****
        ELSE IF (RS3(1).EQ.(0.000)) THEN
            RS3(0) = 1.000
        ELSE
            RS3(0) = 0.000
        ENDIF
*****
* The following calculates the total number of rise/set events *
* between the ground station and the three satellites or between *
* the three satellites. *
*****
        TOTAL = TOT1+TOT2+TOT3

        DO 47 COUNT = 1,500
            READ(60,*,END=48)EVENT(COUNT)
        47 CONTINUE
        48 CONTINUE
*****
* If there were no rise/set times at all, the following message is *
* written to the final output file and the program ends. *
*****
        IF (COUNT.EQ.1) THEN
            WRITE(25,*)'There were no opportunities for viewing
+ with this configuration.'
            GOTO 215
        ENDIF
*****
* The following performs a sort to place the list of all events in *
* order from smallest number to largest. *
*****
        DO 60 COUNT = 1,TOTAL-1
            DO 50 X = 1,TOTAL-1
                IF (EVENT(X+1).LT.EVENT(X)) THEN
                    TEMP = EVENT(X+1)
                    EVENT(X+1) = EVENT(X)
                    EVENT(X) = TEMP
                ENDIF
            50 CONTINUE
        60 CONTINUE

```

\*\*\*\*\*  
\* The following loops determine the intermediate status of v'ibil-  
\* ity at each event. \*

\*\*\*\*\*

```
DO 80 COUNT = 0, TOTAL
  IF (TOT1.EQ.0) THEN
    TT(COUNT,1) = RS1(0)
    GOTO 80
  ENDIF
DO 70 IN = 0, TOT1 - 1
  IF (EVENT(COUNT).GE.E1(IN).AND.EVENT(COUNT).LT.E1(IN+1)) THEN
    TT(COUNT,1) = RS1(IN)
    GOTO 80
  ELSE IF (EVENT(COUNT).EQ.E1(IN+1)) THEN
    TT(COUNT,1) = RS1(IN+1)
    GOTO 80
  ELSE IF (EVENT(COUNT).GT.E1(IN+1).AND.IN.EQ.(TOT1-1)) THEN
    TT(COUNT,1) = RS1(TOT1)
    GOTO 80
  ENDIF
70 CONTINUE
80 CONTINUE
```

```
DO 100 COUNT = 0, TOTAL
  IF (TOT2.EQ.0) THEN
    TT(COUNT,2) = RS2(0)
    GOTO 100
  ENDIF
DO 90 IN = 0, TOT2 - 1
  IF (EVENT(COUNT).GE.E2(IN).AND.EVENT(COUNT).LT.E2(IN+1)) THEN
    TT(COUNT,2) = RS2(IN)
    GOTO 100
  ELSE IF (EVENT(COUNT).EQ.E2(IN+1)) THEN
    TT(COUNT,2) = RS2(IN+1)
    GOTO 100
  ELSE IF (EVENT(COUNT).GT.E2(IN+1).AND.IN.EQ.(TOT2-1)) THEN
    TT(COUNT,2) = RS2(TOT2)
    GOTO 100
  ENDIF
90 CONTINUE
100 CONTINUE
```

```
DO 120 COUNT = 0, TOTAL
DO 110 IN = 0, TOT3 - 1
  IF (EVENT(COUNT).GE.E3(IN).AND.EVENT(COUNT).LT.E3(IN+1)) THEN
    TT(COUNT,3) = RS3(IN)
    GOTO 120
  ELSE IF (EVENT(COUNT).EQ.E3(IN+1)) THEN
    TT(COUNT,3) = RS3(IN+1)
    GOTO 120
  ELSE IF (EVENT(COUNT).GT.E3(IN+1).AND.IN.EQ.(TOT3-1)) THEN
    TT(COUNT,3) = RS3(TOT3)
    GOTO 120
  ENDIF
110 CONTINUE
120 CONTINUE
```

CHANGE = 1

\*\*\*\*\*

\* The following determines if continuous communication is possible.\*  
\* For 3 satellites, the sum must be greater than or equal to 2. \*  
\* For a ground station and three satellites, the sum must be \*  
\* greater than or equal to 1. \*

\*\*\*\*\*

```
IF (KIND.EQ.1) THEN
```

DO 130 COUNT = 0, TOTAL

```
IF (SUM.LE.(1.000).AND.CHANGE.EQ.1) THEN  
WRITE(25,200)EVENT(COUNT)  
CHANGE = 0  
ENDIF
```

```
IF (SUM.GE.(2.000).AND.CHANGE.EQ.0) THEN  
WRITE(25,210)EVENT(COUNT)  
CHANGE = 1  
ENDIF
```

130 CONTINUE

```
ELSE IF (KIND.EQ.0) THEN  
DO 140 COUNT = 0,TOTAL  
SUM = TT(COUNT,1) + TT(COUNT,2) + TT(COUNT,3)  
IF (SUM.LT.(1.000).AND.CHANGE.EQ.1) THEN  
WRITE(25,200)EVENT(COUNT)  
CHANGE = 0  
ENDIF
```

```
IF (SUM.GE.(1.000).AND.CHANGE.EQ.0) THEN  
WRITE(25,210)EVENT(COUNT)  
CHANGE = 1  
ENDIF
```

140 CONTINUE  
ENDIF

215 CONTINUE

```
CLOSE(UNIT=STATE,STATUS='DELETE')  
CLOSE(UNIT=STATE+1,STATUS='DELETE')  
CLOSE(UNIT=STATE+2,STATUS='DELETE')  
CLOSE(UNIT=60)  
CLOSE(UNIT=10,STATUS='DELETE')  
CLOSE(UNIT=11,STATUS='DELETE')  
CLOSE(UNIT=12,STATUS='DELETE')
```

200 FORMAT('Communications broken at ',F12.5,' SEC')  
210 FORMAT('Communications re-established at ',F12.5,' SEC')

RETURN

END

\*

```

*          SUBROUTINE CVC4          *
*
* PURPOSE:  The purpose of this subroutine is to accomplish a
*           check for constant visibility between 4 satellites.
*           For visibility to be maintained, at least 3
*           paths of visibility must be maintained at all times.
*
* VARIABLES:
* INPUT:
*   STATE   - File number specifier
*
* LOCAL VARIABLES:
* CHANGE   - specifies when communication is broken or re-
*           established
* COUNT    - do loop counter
* E1,...,E6 - Specifies rise/set time for visibility function
*           between 6 satellites
* EVENT    - Array of all rise/set times
* IN       - do loop counter
* RS1,...RS6 - 1 if the event is a rise; 0 if the event is a set
* SUM      - Sum of RS1...RS6 at a given event
* T        - time zero read from file of visibility data
*           - in this subroutine, T serves only as a place
*           holder
* TEMP     - Temporary value used in sort loop
* TOTAL    - Total number of events
* TOT1     - Number of events between Satellite #1 and
*           Satellite #2
* TOT2     - Number of events between Satellite #1 and
*           Satellite #3
* TOT3     - Number of events between Satellite #1 and
*           Satellite #4
* TOT4     - Number of events between Satellite #2 and
*           Satellite #3
* TOT5     - Number of events between Satellite #2 and
*           Satellite #4
* TOT6     - Number of events between Satellite #3 and
*           Satellite #4
* TT       - Holds all event times and the corresponding
*           status of the visibility function for each
*           viewing path
* V        - value of the chosen visibility function at time
*           equal to zero
* X        - do loop counter

```

```
*****
```

```
SUBROUTINE CVC4(STATE)
```

```

REAL*8  E1(0:50),E2(0:50),E3(0:50),E4(0:50),E5(0:50)
REAL*8  E6(0:50),EVENT(0:200),RS1(0:50),RS2(0:50),RS3(0:50)
REAL*8  RS4(0:50),RS5(0:50),RS6(0:50),SUM,T,TEMP,TT(0:200,6),V
INTEGER CHANGE,COUNT,IN,STATE,TOTAL,TOT1,TOT2,TOT3,TOT4,TOT5
INTEGER TOT6,X

```

```

WRITE(*,*)
WRITE(*,*)'The computer is in the process of determining'
WRITE(*,*)'whether or not constant communications is '
WRITE(*,*)'maintained. When the program has finished'
WRITE(*,*)'running, results may be found in RISESET.DAT.'

```

```

OPEN(UNIT=41,FILE='STAT1.DAT',STATUS='OLD')
OPEN(UNIT=42,FILE='STAT2.DAT',STATUS='OLD')
OPEN(UNIT=43,FILE='STAT3.DAT',STATUS='OLD')
OPEN(UNIT=44,FILE='STAT4.DAT',STATUS='OLD')

```

```
OPEN(UNIT=10,FILE='VIS1.DAT',STATUS='OLD')
OPEN(UNIT=11,FILE='VIS2.DAT',STATUS='OLD')
OPEN(UNIT=12,FILE='VIS3.DAT',STATUS='OLD')
OPEN(UNIT=13,FILE='VIS4.DAT',STATUS='OLD')
OPEN(UNIT=14,FILE='VIS5.DAT',STATUS='OLD')
OPEN(UNIT=15,FILE='VIS6.DAT',STATUS='OLD')
```

```
*****
*           SATELLITE #1 AND SATELLITE #2           *
*****
```

```
TOT1 = 0
DO 20 COUNT = 1,100
  READ(STATE,*,END=25)E1(COUNT),RS1(COUNT)
  TOT1 = COUNT
20 CONTINUE
25 E1(0) = 0.000
```

```
IF (TOT1.EQ.0) THEN
  READ(10,*)T,V
  IF (V.GT.(0.000)) THEN
    RS1(0)=1.000
  ELSE
    RS1(0)=0.000
  ENDIF
ELSE IF (RS1(1).EQ.(0.000)) THEN
  RS1(0) = 1.000
ELSE
  RS1(0) = 0.000
ENDIF
```

```
*****
*           SATELLITE #1 AND SATELLITE #3           *
*****
```

```
TOT2 = 0
DO 30 COUNT = 1,100
  READ(STATE+1,*,END=35)E2(COUNT),RS2(COUNT)
  TOT2 = COUNT
30 CONTINUE
35 E2(0) = 0.000
```

```
IF (TOT2.EQ.0) THEN
  READ(11,*)T,V
  IF (V.GT.(0.000)) THEN
    RS2(0)=1.000
  ELSE
    RS2(0)=0.000
  ENDIF
ELSE IF (RS2(1).EQ.(0.000)) THEN
  RS2(0) = 1.000
ELSE
  RS2(0) = 0.000
ENDIF
```

```
*****
*           SATELLITE #1 AND SATELLITE #4           *
*****
```

```
TOT3 = 0
DO 40 COUNT = 1,100
  READ(STATE+2,*,END=45)E3(COUNT),RS3(COUNT)
  TOT3 = COUNT
40 CONTINUE
45 E3(0) = 0.000
```

```
IF (TOT3.EQ.0) THEN
  READ(12,*)T,V
```

```

ENDIF
*****
* The following line checks the first condition of the first rise *
* or set time then assigns the appropriate condition to time zero. *
*****
ELSE IF (RS3(1).EQ.(0.000)) THEN
  RS3(0) = 1.000
ELSE
  RS3(0) = 0.000
ENDIF
*****
* SATELLITE #2 AND SATELLITE #3 *
*****
TOT4 = 0
DO 50 COUNT = 1,100
  READ(STATE+3,*,END=55)E4(COUNT),RS4(COUNT)
  TOT4 = COUNT
50 CONTINUE
55 E4(0) = 0.000

IF (TOT4.EQ.0) THEN
  READ(13,*)T,V
  IF (V.GT.(0.000)) THEN
    RS4(0)=1.000
  ELSE
    RS4(0)=0.000
  ENDIF
*****
* The following line checks the first condition of the first rise *
* or set time then assigns the appropriate condition to time zero. *
*****
ELSE IF (RS4(1).EQ.(0.000)) THEN
  RS4(0) = 1.000
ELSE
  RS4(0) = 0.000
ENDIF
*****
* SATELLITE #2 AND SATELLITE #4 *
*****
TOT5 = 0
DO 60 COUNT = 1,100
  READ(STATE+4,*,END=65)E5(COUNT),RS5(COUNT)
  TOT5 = COUNT
60 CONTINUE
65 E5(0) = 0.000

IF (TOT5.EQ.0) THEN
  READ(14,*)T,V
  IF (V.GT.(0.000)) THEN
    RS5(0)=1.000
  ELSE
    RS5(0)=0.000
  ENDIF
*****
* The following line checks the first condition of the first rise *
* or set time then assigns the appropriate condition to time zero. *
*****
ELSE IF (RS5(1).EQ.(0.000)) THEN
  RS5(0) = 1.000
ELSE
  RS5(0) = 0.000
ENDIF

```



```

DO 70 COUNT = 1,100
  READ(STATE+5,*,END=73)E6(COUNT),RS6(COUNT)
  TOT6 = COUNT
70 CONTINUE
73 E6(0) = 0.000

```

```

IF (TOT6.EQ.0) THEN
  READ(15,*)T,V
  IF (V.GT.(0.000)) THEN
    RS6(0)=1.000
  ELSE
    RS6(0)=0.000
  ENDIF

```

```

*****
* The following line checks the first condition of the first rise *
* or set time then assigns the appropriate condition to time zero. *
*****

```

```

ELSE IF (RS6(1).EQ.(0.000)) THEN
  RS6(0) = 1.000
ELSE
  RS6(0) = 0.000
ENDIF

```

```

*****
* The following calculates the total number of rise/set events *
* between the ground station and the three satellites. *
*****

```

```

TOTAL = TOT1+TOT2+TOT3+TOT4+TOT5+TOT6

```

```

DO 75 COUNT = 1,500
  READ(60,*,END=76)EVENT(COUNT)
75 CONTINUE
76 CONTINUE

```

```

*****
* If there were no rise/set times at all, the following message is *
* written to the final output file and the program ends. *
*****

```

```

IF (COUNT.EQ.1) THEN
  WRITE(25,*)'There were no opportunities for viewing with this
+ configuration.'
  GOTO 215
ENDIF

```

```

*****
* The following performs a sort to place the list of all events in *
* order from smallest number to largest. *
*****

```

```

DO 80 COUNT = 1,TOTAL-1
  DO 78 X = 1,TOTAL-1
    IF (EVENT(X+1).LT.EVENT(X)) THEN
      TEMP = EVENT(X+1)
      EVENT(X+1) = EVENT(X)
      EVENT(X) = TEMP
    ENDIF
  ENDIF

```

```

78 CONTINUE
80 CONTINUE

```

```

EVENT(0) = 0.000

```

```

*****
* The following loops determine the intermediate status of visibil- *
* ity at each event. *
*****

```

```

DO 100 COUNT = 0,TOTAL
  IF (TOT1.EQ.0) THEN

```

```

        TT(COUNT,1) = RS1(IN)
        GOTO 100
    ELSE IF (EVENT(COUNT).EQ.E1(IN+1)) THEN
        TT(COUNT,1) = RS1(IN+1)
        GOTO 100
    ELSE IF (EVENT(COUNT).GT.E1(IN).AND.IN.EQ.(TOT1-1)) THEN
        TT(COUNT,1) = RS1(TOT1)
        GOTO 100
    ENDIF
90 CONTINUE
100 CONTINUE

DO 120 COUNT = 0,TOTAL
    IF (TOT2.EQ.0) THEN
        TT(COUNT,2) = RS2(0)
        GOTO 120
    ENDIF
DO 110 IN = 0,TOT2 - 1
    IF (EVENT(COUNT).GE.E2(IN).AND.EVENT(COUNT).LT.E2(IN+1)) THEN
        TT(COUNT,2) = RS2(IN)
        GOTO 120
    ELSE IF (EVENT(COUNT).EQ.E2(IN+1)) THEN
        TT(COUNT,2) = RS2(IN+1)
        GOTO 120
    ELSE IF (EVENT(COUNT).GT.E2(IN).AND.IN.EQ.(TOT2-1)) THEN
        TT(COUNT,2) = RS2(TOT2)
        GOTO 120
    ENDIF
110 CONTINUE
120 CONTINUE

DO 140 COUNT = 0,TOTAL
    IF (TOT3.EQ.0) THEN
        TT(COUNT,3) = RS3(0)
        GOTO 140
    ENDIF
DO 130 IN = 0,TOT3 - 1
    IF (EVENT(COUNT).GE.E3(IN).AND.EVENT(COUNT).LT.E3(IN+1)) THEN
        TT(COUNT,3) = RS3(IN)
        GOTO 140
    ELSE IF (EVENT(COUNT).EQ.E3(IN+1)) THEN
        TT(COUNT,3) = RS3(IN+1)
        GOTO 140
    ELSE IF (EVENT(COUNT).GT.E3(IN).AND.IN.EQ.(TOT3-1)) THEN
        TT(COUNT,3) = RS3(TOT3)
        GOTO 140
    ENDIF
130 CONTINUE
140 CONTINUE

DO 160 COUNT = 0,TOTAL
    IF (TOT4.EQ.0) THEN
        TT(COUNT,4) = RS4(0)
        GOTO 160
    ENDIF
DO 150 IN = 0,TOT4 - 1
    IF (EVENT(COUNT).GE.E4(IN).AND.EVENT(COUNT).LT.E4(IN+1)) THEN
        TT(COUNT,4) = RS4(IN)
        GOTO 160
    ELSE IF (EVENT(COUNT).EQ.E4(IN+1)) THEN
        TT(COUNT,4) = RS4(IN+1)

```

160 CONTINUE  
160 CONTINUE

```
DO 180 COUNT = 0, TOTAL
  IF (TOT5.EQ.0) THEN
    TT(COUNT,5) = RS5(0)
    GOTO 180
  ENDIF
DO 170 IN = 0, TOT5 - 1
  IF (EVENT(COUNT).GE.E5(IN).AND.EVENT(COUNT).LT.E5(IN+1)) THEN
    TT(COUNT,5) = RS5(IN)
    GOTO 180
  ELSE IF (EVENT(COUNT).EQ.E5(IN+1)) THEN
    TT(COUNT,5) = RS5(IN+1)
    GOTO 180
  ELSE IF (EVENT(COUNT).GT.E5(IN).AND.IN.EQ.(TOT5-1)) THEN
    TT(COUNT,5) = RS5(TOT5)
    GOTO 180
  ENDIF
170 CONTINUE
180 CONTINUE
```

```
DO 200 COUNT = 0, TOTAL
  IF (TOT6.EQ.0) THEN
    TT(COUNT,6) = RS6(0)
    GOTO 200
  ENDIF
DO 190 IN = 0, TOT6 - 1
  IF (EVENT(COUNT).GE.E6(IN).AND.EVENT(COUNT).LT.E6(IN+1)) THEN
    TT(COUNT,6) = RS6(IN)
    GOTO 200
  ELSE IF (EVENT(COUNT).EQ.E6(IN+1)) THEN
    TT(COUNT,6) = RS6(IN+1)
    GOTO 200
  ELSE IF (EVENT(COUNT).GT.E6(IN).AND.IN.EQ.(TOT6-1)) THEN
    TT(COUNT,6) = RS6(TOT6)
    GOTO 200
  ENDIF
190 CONTINUE
200 CONTINUE
```

CHANGE = 1

```
*****
* The following determines if continuous communication is possible.*
* For 4 satellites, the sum must be greater than or equal to 3. *
*****
```

```
DO 210 COUNT = 0, TOTAL
  SUM = TT(COUNT,1) + TT(COUNT,2) + TT(COUNT,3) + TT(COUNT,4) +
  + TT(COUNT,5) + TT(COUNT,6)
  IF (SUM.LT.(3.000).AND.CHANGE.EQ.1) THEN
    WRITE(25,220)EVENT(COUNT)
    CHANGE = 0
  ENDIF

  IF (SUM.GE.(3.000).AND.CHANGE.EQ.0) THEN
    WRITE(25,230)EVENT(COUNT)
    CHANGE = 1
  ENDIF
```

210 CONTINUE  
215 CONTINUE

CLOSE(UNIT=STATE+5,STATUS='DELETE')  
CLOSE(UNIT=60)

CLOSE(UNIT=10,STATUS='DELETE')  
CLOSE(UNIT=11,STATUS='DELETE')  
CLOSE(UNIT=12,STATUS='DELETE')  
CLOSE(UNIT=13,STATUS='DELETE')  
CLOSE(UNIT=14,STATUS='DELETE')  
CLOSE(UNIT=15,STATUS='DELETE')

220 FORMAT('Communications broken at ',F12.5,' SEC')  
230 FORMAT('Communications re-established at ',F12.5,' SEC')

RETURN  
END

```

*****
*                               SUBROUTINE GSSVC4                               *
*   (Ground Station Satellite Continuous Visibility Check (4))                 *
*   PURPOSE: The purpose of this subroutine is to accomplish a               *
*             check for constant visibility between a ground station         *
*             and 3 satellites.                                               *
*                                                                              *
*   VARIABLES:                                                                *
*   INPUT:                                                                    *
*     STATE      - File number specifier                                     *
*                                                                              *
*   LOCAL VARIABLES:                                                         *
*     CHANGE     - specifies when communication is broken or re-             *
*                 established                                                *
*     COUNT      - do loop counter                                           *
*     E1,...,E4  - Specifies rise/set time for visibility function           *
*                 a ground station and 4 satellites                         *
*     EVENT      - Array of all rise/set times                               *
*     IN         - do loop counter                                           *
*     RS1,..,RS4 - 1 if the event is a rise; 0 if the event is a set       *
*     SUM        - Sum of RS1,...,RS4 at a given event                       *
*     T          - time zero of any selected visibility function             *
*                 - in this subroutine T is a place holder                  *
*     TEMP       - Temporary value used in sort loop                         *
*     TOTAL      - Total number of events                                    *
*     TOT1       - Number of events between Ground Station #? and          *
*                 Satellite =1                                              *
*     TOT2       - Number of events between Ground Station #? and          *
*                 Satellite =2                                              *
*     TOT3       - Number of events between Ground Station #? and          *
*                 Satellite =3                                              *
*     TOT4       - Number of events between Ground Station #? and          *
*                 Satellite =4                                              *
*     TT         - Holds all event times and the corresponding              *
*                 status of the visibility function for each                *
*                 viewing path                                              *
*     V          - the first value of the selected visibility                *
*                 function                                                  *
*     X          - do loop counter                                           *

```

```
*****
```

```
SUBROUTINE GSSVC4(STATE)
```

```

REAL*8  E1(0:100),E2(0:100),E3(0:100),E4(0:100),EVENT(0:500)
REAL*8  RS1(0:100),RS2(0:100),RS3(0:100),RS4(0:100),T,TEMP
REAL*8  TT(0:500,4),V
INTEGER CHANGE,CHECK,COUNT,IN,STATE,SUM
INTEGER TOTAL,TOT1,TOT2,TOT3,TOT4,X

```

```

OPEN(UNIT=41,FILE='STAT1.DAT',STATUS='OLD')
OPEN(UNIT=42,FILE='STAT2.DAT',STATUS='OLD')
OPEN(UNIT=43,FILE='STAT3.DAT',STATUS='OLD')
OPEN(UNIT=44,FILE='STAT4.DAT',STATUS='OLD')
OPEN(UNIT=60,FILE='TRACKER.DAT',STATUS='OLD')

```

```
REWIND(UNIT=60)
```

```

OPEN(UNIT=10,FILE='VIS1.DAT',STATUS='OLD')
OPEN(UNIT=11,FILE='VIS2.DAT',STATUS='OLD')
OPEN(UNIT=12,FILE='VIS3.DAT',STATUS='OLD')
OPEN(UNIT=13,FILE='VIS4.DAT',STATUS='OLD')

```

```
*****
```

```
DO 20 COUNT = 1,100
  READ(STATE,*,END=25)E1(COUNT),RS1(COUNT)
  TOT1 = COUNT
20 CONTINUE
25 E1(0) = 0.000
```

```
IF (TOT1.EQ.0) THEN
  READ(10,*)T,V
  IF (V.GT.(0.000)) THEN
    RS1(0) = 1.000
  ELSE
    RS1(0) = 0.000
  ENDIF
ELSE IF (RS1(1).EQ.(0.000)) THEN
  RS1(0) = 1.000
ELSE
  RS1(0) = 0.000
ENDIF
```

```
*****
*                GROUND STATION AND SATELLITE #2                *
*****
```

```
TOT2 = 0
DO 30 COUNT = 1,100
  READ(STATE+1,*,END=35)E2(COUNT),RS2(COUNT)
  TOT2 = COUNT
30 CONTINUE
35 E2(0) = 0.000
```

```
IF (TOT2.EQ.0) THEN
  READ(11,*)T,V
  IF (V.GT.(0.000)) THEN
    RS2(0) = 1.000
  ELSE
    RS2(0) = 0.000
  ENDIF
ELSE IF (RS2(1).EQ.(0.000)) THEN
  RS2(0) = 1.000
ELSE
  RS2(0) = 0.000
ENDIF
```

```
*****
*                GROUND STATION AND SATELLITE #3                *
*****
```

```
TOT3 = 0
DO 40 COUNT = 1,100
  READ(STATE+2,*,END=45)E3(COUNT),RS3(COUNT)
  TOT3 = COUNT
40 CONTINUE
45 E3(0) = 0.000
```

```
IF (TOT3.EQ.0) THEN
  READ(12,*)T,V
  IF (V.GT.(0.000)) THEN
    RS3(0) = 1.000
  ELSE
    RS3(0) = 0.000
  ENDIF
ELSE IF (RS3(1).EQ.(0.000)) THEN
  RS3(0) = 1.000
ELSE
  RS3(0) = 0.000
ENDIF
```

```
*****
```

```

DO 50 COUNT = 1,100
  READ(STATE+3,*,END=53)E4(COUNT),RS4(COUNT)
  TOT4 = COUNT
50 CONTINUE
53 E4(0) = 0.0D0

```

```

IF (TOT4.EQ.0) THEN
  READ(13,*)T,V
  IF (V.GT.(0.0D0)) THEN
    RS4(0) = 1.0D0
  ELSE
    RS4(0) = 0.0D0
  ENDIF
ELSE IF (RS4(1).EQ.(0.0D0)) THEN
  RS4(0) = 1.0D0
ELSE
  RS4(0) = 0.0D0
ENDIF

```

```

*****
* The following calculates the total number of events between the *
* ground station and four satellites. *
*****

```

```

TOTAL = TOT1+TOT2+TOT3+TOT4
CHECK = 0
DO 55 COUNT = 1,500
  READ(60,*,END=58)EVENT(COUNT)
  CHECK = COUNT
55 CONTINUE
58 CONTINUE

```

```

IF (CHECK.EQ.0) THEN
  WRITE(25,*)'There were no visibility opportunities with this
+ configuration.'
  GOTO 160
ENDIF

```

```

*****
* The following orders all the events from the smallest to the *
* largest. *
*****

```

```

DO 65 COUNT = 1,TOTAL-1
  DO 60 X = 1,TOTAL-1
    IF (EVENT(X+1).LT.EVENT(X)) THEN
      TEMP = EVENT(X+1)
      EVENT(X+1) = EVENT(X)
      EVENT(X) = TEMP
    ENDIF
  60 CONTINUE
65 CONTINUE

```

```

EVENT(0) = 0.0D0

```

```

*****
* The following determines the status of communications for each *
* possible visibility path at each event. *
*****

```

```

DO 80 COUNT = 0,TOTAL
  IF (TOT1.EQ.0) THEN
    TT(COUNT,1) = RS1(0)
    GOTO 80
  ENDIF
  DO 70 IN = 0,TOT1 - 1
    IF (EVENT(COUNT).GE.E1(IN).AND.EVENT(COUNT).LT.E1(IN+1)) THEN
      TT(COUNT,1) = RS1(IN)
      GOTO 80
    ENDIF
  70 CONTINUE
80 CONTINUE

```

```

        GOTO 80
    ELSE IF (EVENT(COUNT).GT.E1(IN+1).AND.IN.EQ.(TOT1-1)) THEN
        TT(COUNT,1) = RS1(IN+1)
        GOTO 80
    ENDIF
70 CONTINUE
80 CONTINUE

DO 100 COUNT = 0,TOTAL
    IF (TOT2.EQ.0) THEN
        TT(COUNT,2) = RS2(0)
        GOTO 100
    ENDIF
DO 90 IN = 0,TOT2 - 1
    IF (EVENT(COUNT).GE.E2(IN).AND.EVENT(COUNT).LT.E2(IN+1)) THEN
        TT(COUNT,2) = RS2(IN)
        GOTO 100
    ELSE IF (EVENT(COUNT).EQ.E2(IN+1)) THEN
        TT(COUNT,2) = RS2(IN+1)
        GOTO 100
    ELSE IF (EVENT(COUNT).GT.E2(IN+1).AND.IN.EQ.(TOT2-1)) THEN
        TT(COUNT,2) = RS2(IN+1)
        GOTO 100
    ENDIF
90 CONTINUE
100 CONTINUE

DO 120 COUNT = 0,TOTAL
    IF (TOT3.EQ.0) THEN
        TT(COUNT,3) = RS3(0)
        GOTO 120
    ENDIF
DO 110 IN = 0,TOT3 - 1
    IF (EVENT(COUNT).GE.E3(IN).AND.EVENT(COUNT).LT.E3(IN+1)) THEN
        TT(COUNT,3) = RS3(IN)
        GOTO 120
    ELSE IF (EVENT(COUNT).EQ.E3(IN+1)) THEN
        TT(COUNT,3) = RS3(IN+1)
        GOTO 120
    ELSE IF (EVENT(COUNT).GT.E3(IN+1).AND.IN.EQ.(TOT3-1)) THEN
        TT(COUNT,3) = RS3(IN+1)
        GOTO 120
    ENDIF
110 CONTINUE
120 CONTINUE

DO 140 COUNT = 0,TOTAL
    IF (TOT4.EQ.0) THEN
        TT(COUNT,4) = RS4(0)
        GOTO 140
    ENDIF
DO 130 IN = 0,TOT4 - 1
    IF (EVENT(COUNT).GE.E4(IN).AND.EVENT(COUNT).LT.E4(IN+1)) THEN
        TT(COUNT,4) = RS4(IN)
        GOTO 140
    ELSE IF (EVENT(COUNT).EQ.E4(IN+1)) THEN
        TT(COUNT,4) = RS4(IN+1)
        GOTO 140
    ELSE IF (EVENT(COUNT).GT.E4(IN+1).AND.IN.EQ.(TOT4-1)) THEN
        TT(COUNT,4) = RS4(IN+1)
        GOTO 140
    ENDIF
130 CONTINUE
140 CONTINUE

```



```
*****
* The next DO loop determines if constant communication is *
* possible. If the sum is equal to zero, communication is not *
* possible. If the sum is greater than or equal to one, communi- *
* cation is possible. *
*****
```

```
DO 150 COUNT = 0, TOTAL
  SUM = TT(COUNT,1) + TT(COUNT,2) + TT(COUNT,3) + TT(COUNT,4)
  IF (SUM.EQ.(0.0D0).AND.CHANGE.EQ.1) THEN
    WRITE(25,*)'Communications broken at ',EVENT(COUNT)
    CHANGE = 0
  ENDIF

  IF (SUM.GE.(1.0D0).AND.CHANGE.EQ.0) THEN
    WRITE(25,*)'Communications re-established at ',EVENT(COUNT)
    CHANGE = 1
  ENDIF
```

```
150 CONTINUE
```

```
160 CONTINUE
```

```
CLOSE(UNIT=STATE,STATUS='DELETE')
CLOSE(UNIT=STATE+1,STATUS='DELETE')
CLOSE(UNIT=STATE+2,STATUS='DELETE')
CLOSE(UNIT=STATE+3,STATUS='DELETE')
CLOSE(UNIT=60)
CLOSE(UNIT=10,STATUS='DELETE')
CLOSE(UNIT=11,STATUS='DELETE')
CLOSE(UNIT=12,STATUS='DELETE')
CLOSE(UNIT=13,STATUS='DELETE')
```

```
RETURN
```

```
END
```

**END  
FILMED**

**DATE:**

*1-92*

**DTIC**