DEVELOPING SITUATIONAL CONDITIONS AND PROGRAM CODES FOR PARALLEL SITUATIONAL ANALYSIS SOLVER BASED ON CONIC UMBRA/SUNLIT MODELS

Atanas Atanassov

Space Research and Technology Institute – Bulgarian Academy of Sciences
e-mail: At_M_Atanassov@yahoo.com

Keywords: Space mission analysis and design, Situational analysis, Constraints analysis, Situational problem, Situational conditions; Parallel algorithms

Abstract
The search for optimal time intervals for satellite operations performance is based on verifying specific conditions of geometric or physical nature. Several conditions are combined in a situational problem. To be able to solve different situational problems, it is necessary to develop program codes for verifying different situational conditions in advance. Two situational conditions for determining if the satellite is in the sunlit zone or the umbra based on the conical models of the earth's shadow are presented. Necessary (but not sufficient) conditions are introduced to locate the satellite in the umbra or the sunlit zone, respectively. These conditions simplify the calculations and increase the computational efficiency for a large part of the satellite orbit. Problems with one or more conditions are solved by a developed parallel solver for situational analysis. Conditions checking the position of a satellite relative to the Earth’s shadow can be combined with other situational conditions. A model of description of situational conditions is shown.

Introduction
Current trends in the development of satellite technologies lead on the one hand to the use of smaller satellites and on the other hand, to multi-satellite missions instead of large multi-purpose satellites [1, 2] The process of space missions’ analysis and design is related to the clarification and optimization of many and different parameters linked to scientific instruments, satellite subsystems, orbital parameters or templates of multi-satellite systems.

Computer simulations are playing an increasing role at all stages of preparation and operational implementation of multi-satellite missions. The application of parallel algorithms and calculations is crucial in solving large-scale problems involving hundreds and thousands of satellites.

An important part of the space missions’ analysis is the so-called situational analysis. The situational analysis deals with the determination of optimal time
intervals suitable for the execution of satellite operations, depending on various geometric and physical conditions. This type of analysis is applied to different stages of mission preparation—starting with the conceptual study and preliminary analysis, going through mission definition, design and development, and finishing with implementation. The determination of the initial and final moments of the passage of a satellite through the shadow of the Earth as well as through the sunlit zone has wide practical applications. The evaluation of the energy produced by solar panels and its use in the execution of various operations is an example of such an application. These moments are necessary to perform measurements in the Earth's shadow or in the illuminated part of a satellite's orbit.

Wertz and Larson [3] emphasize that mission analysis algorithms must be simple and effective enough to allow multiple runs, collect statistical data, and explore various scenarios and design options. The development of effective methods and computer programs for situational analysis is very important when many problems need to be solved, each involving more than one situation.

Algorithms and calculation tools for space mission analysis and design are under development at the Space Research and Technology Institute at the Bulgarian Academy of Sciences. One such tool is a parallel solver for situational analysis [4]. Algorithms and program realization of the conical model of the Earth’s shadow are discussed in the present work. Algorithms and program codes suitable for multi-satellite situational analysis are proposed.

**Concept of situation analysis**

As pointed out above, situational analysis deals with solving problems related to satellite operation optimization according to necessary restrictive conditions. Each situational problem SP is composed of one or a conjunction of several situational conditions $sc_i$ of different types:

$$ SP = sc_1 \land sc_2 \land ... \land sc_n $$

The conditions themselves are predicate functions accepting values of one or zero. They can be generally represented as:

$$ sc_i = sc_i(\{\vec{R}\}, \{\alpha\}, \{\beta\}, t) $$

where $\{\vec{R}\}$ is a set of radius vectors of objects, $\{\alpha\}$ is a set of parameters of the mathematical model describing the situational condition and $\{\beta\}$ is a set of constraints that are specific to the conditions. The calculation of the $SP$ function is done by sequentially checking the conditions $sc_i$. The application of Horner’s rule allows cancelling the verification of the conditions when an unfulfilled condition $sc_l$ is found:

$$ SP = ( ... (... (sc_1 \land sc_2) \land ... ) \land sc_l ... \land sc_{n-1} ) \land sc_n $$

16
It is important to keep in mind that each condition $sc_i$ is satisfied within the time interval $T_{i,k}$ and is not fulfilled for the next adjacent interval $T_{i,k}^{*}$. The only requirement to $T_{i,k}$ and $T_{i,k}^{*}$, which will be taken into account, are $T_{i,k}, T_{i,k}^{*} \gg \Delta t$, where $\Delta t$ is a step in simulation time. The index $k$ denotes the serial number of the interval within the time horizon.

**Parallel solving of situational problems**

A simulation of multi-satellite missions with many objects of observation requires solving many situational problems. Repeated simulations to optimize different parameters also lead to the need for multiple situational analyses. The functionality and reliability of multi-satellite systems are a result of the space environment’s impact on the subsystems and the scientific instruments over a long time horizon. The computer simulations clarifying changes in the functionality and the reliability require computational efficiency of situational analysis. The application of parallel calculations is a step in the right direction.

Advances in computer technologies provide new opportunities for the compilation of even more complex simulation models but also place demands on their development. The article comments on the solving of situational problems in the context of parallel calculations (using multiprocessor and multicore systems). This requires special approaches and program models in the development of computational algorithms. The reasons for applying parallel calculations are the following:

1. complex mathematical models are used for the calculation of different situational conditions when searching for orbital events;
2. simultaneous solving of many situation problems is included in a simulation model;
3. the simulation time intervals $(t_{\text{begin}}, t_{\text{end}})$ are large;
4. Repeatedly solving of situation model based on situational analysis for space mission parameters optimization (concerning satellites, instruments, ground-based stations’ and objects of investigations).

All compiled situational problems can be applied to all members in a multi-satellite system. The number of situational problems needed to be solved may increase according to the number of satellites and objects for observation and scientific problems for investigation.

Different situational conditions based on specific computational models give rise to a different number of computational operations. Such models are irregular and an imbalance occurs when the calculations are parallelized. This is due to the uneven distribution of computational operations between the available processors. In case of poor distribution, some processors complete their problems and are free,
while others continue their calculations. The “Pool of threads” model copes with this problem [5].

**Parallel Situational Analysis Solver**

A parallel solver for situational analysis was developed for this purpose. It is a processing program that consistently checks the feasibility of the conditions in a particular situational problem. The parallelization is based on computational threads organized in a variant of the program model "pool of threads" [6]. In this variant, the threads are synchronized with each other while they receive situational problems to solve (race condition). This excludes the solution of one situational problem by more than one thread. The threads are also synchronized with the parent thread, which initiates the calculations and waits for them to complete at each step of the simulated time. Each thread takes one or more situational problems for processing, according to the specified value of the parameter known as granularity.

**Situational problems designer**

A dialogue editor of situational problems is developed as an auxiliary tool. The compilation of a situational problem is initiated, situational conditions are successively selected, and their respective parameters and restrictions are set with the help of dialogue controls. An already compiled situational problem can be rejected or approved and saved as a template for future use. An optimization method can be selected too. One already assembled situational problem can be related to a particular satellite or all satellites of the respective multi-satellite system.

**Conic Earth shadow model**

Models of the earth's shadow have been developed and used for various purposes long ago. Ferraz-Mello [6] describes the shadow of the Earth with a cylindrical model. Bordovitzina et al. [7] apply a conical model of the Earth's shadow with penumbra. Determining the moments of entry and exit from the shadow is discussed in [8, 9]. Recently Srivastava considers models of Earth's shadow [10]. Here we will present detailed considerations of the conic models (also of the light zone) due to their use in the development of program code for situational analysis purposes.

A geometric model that is used for calculating the parameters of the Earth's shadow is shown in Fig. 1. The circles \((O_s, R_s)\) and \((O_E, R_E)\) represent the Sun and the Earth in a plane passing through the line \(\overrightarrow{T}\) and point S where the satellite is located, respectively. The radii of the Earth \(R_E\) and the Sun \(R_S\) are assumed to be known. The radius vector \(\overrightarrow{R}_{SE}(t)\) of the Sun in a geo-equatorial coordinate system is determined based on methods of astrodynamics [11]. The radius vector of the satellite \(\overrightarrow{r}_{sat}(t)\) is also determined based on numerical or analytical methods [11].
**A first case – umbra**

The lines \(\overline{t}_1\) and \(\overline{t}_2\) are tangent at the points \((T_{S,1}, T_{S,2})\) and \((T_{E,1}, T_{E,2})\) to the Sun and the Earth respectively in a plane defined by the vectors \(\vec{R}_{SE}\) and \(\vec{r}_{sat}\). From the similarity of the triangles \(\Delta O_S T_{S,2} O_H\) and \(\Delta O_E T_{E,2} O_H\) a formula can be derived for the height of the conical shadow of the Earth:

\[
H_E = \frac{R_E |\vec{R}_{SE}|}{R_S - R_E}
\]

**Fig. 1. Sun-Earth’s umbra cone geometry**

In this formula, \(R_E\) denotes the Earth’s radius, the distance between the Sun and the Earth (the magnitude of the vector \(\vec{R}_{SE}\)) varies during the year in the interval from 147 099 760 km to 152 104 285 km. The height of the Earth's shadow \(H_E\) can be assumed to be a slowly changing quantity.

The next step is to determine whether the satellite is in the shadow of the Earth. One condition for this is that the sub-satellite point is located in the unlit part of the earth's surface. It is equivalent to checking if \(\angle O_H O_E T_{E,2} < \angle O_H O_E S\):

\[
\frac{\overline{R}_{SE} |\vec{r}_{sat}|}{|\overline{R}_{SE}| \cdot |\vec{r}_{sat}|} > \frac{O_E T_{E,2}}{H_E},
\]

because it compares the cosines of the indicated angles on both sides of the inequality without the use of arccosines.

To determine whether the satellite is in the shadow of the Earth, it remains to compare the segments \(\overline{S'S''}\) and \(\overline{SS'}\) on the line \(\overline{d}\), perpendicular to the line \(\overline{d}\). From the similarity of the triangles \(\Delta O_H S'S''\) and \(\Delta O_H T_{E,2} O_E\) it follows:
\[
S'S'' = R_E \cdot \frac{H_E - O_E S'}{T_{E,2} O_H} = R_E \cdot \frac{H_E - |\vec{r}_{sat}| \cdot \cos(4O_H O_E S)}{\sqrt{H_E^2 - R_E^2}}
\]

The length of the line segment \(T_{E,2} O_H\), a leg in the right triangle \(\Delta O_H T_{E,2} O_E\), is determined based on the length of the Earth's shadow \(H_E\) and the magnitude of the radius of the Earth \(R_E\) or finally:

\[
S'S'' = R_E \cdot \frac{H_E - \vec{r}_{sat} \cdot \vec{r}_{sat}}{R_E \cdot \vec{r}_{sat}} \cdot \sqrt{H_E^2 - R_E^2}
\]

or

\[
(5) \quad S'S'' = R_E \cdot \frac{H_E - \vec{e}_{SE} \cdot \vec{r}_{sat}}{\sqrt{H_E^2 - R_E^2}}
\]

where \(\vec{e}_{SE}\) is the unit radius vector of the Sun.

The length of \(S'S\) is determined by \(\Delta O_E S'S\):

\[
S'S = |\vec{r}_{sat}| \cdot \sin 4S' O_E S
\]

or

\[
(6) \quad S'S = r_{sat} \cdot \sqrt{1 - \left(\frac{\vec{r}_{SE} \cdot \vec{r}_{sat}}{|\vec{R}_{SE}| \cdot |\vec{r}_{sat}|}\right)^2}
\]

It is important not to forget that determining the length of \(S'S''\) makes sense only if \(O_E S' < H_E\). Otherwise, there is a third case where the satellite is in the so-called umbra.

**Second case – penumbra**

In like manner, we can determine the conditions where a satellite falls in the penumbra of the Earth. The conditions for the penumbra are determined by the satellite-Earth configuration, in which the Sun is partially obscured by the Earth, viewed from the position of the satellite (Figure 2). The lines \(\vec{t}_1'\) and \(\vec{t}_2'\) are tangent at the points \((T_{S,1}', T_{S,2}')\) and \((T_{E,1}', T_{E,2}')\) to the Sun and the Earth respectively in a plane defined by the vectors \(\vec{R}_{SE}\) and \(\vec{r}_{sat}\). From the similarity of the triangles \(\Delta O_S T_{S,2} O_H^*\) and \(\Delta O_E T_{E,2} O_H^*\) it follows:

\[
\frac{H_E^*}{|\vec{R}_{SE}| - H_E^*} = \frac{R_E}{R_S}
\]

or
\( H_E^* = \frac{R_E |\vec{R}_{SZ}|}{R_s + R_E} \)

where \( H_E^* \) denotes the line segment \( \overline{OH} \). \( H_E^* \) is analogous to the height of the conical umbra.

The next step, after determining the distance \( H_E^* \), is to check if a satellite is in the Earth's penumbra. The first condition checks whether the satellite is outside the penumbra (and shadow) zone. For this, it is sufficient to compare the angles \( \angle O_S O_E T_{E,1} \) and \( \angle O_S O_E S \) (or their cosines). Further calculations make sense if \( \angle O_S O_E T_{E,1} < \angle O_S O_E S \):

\[
(8) \quad -\frac{\vec{R}_{SE} \cdot \vec{r}_{sat}}{|\vec{R}_{SE}| \cdot |\vec{r}_{sat}|} > \frac{\overline{OE} T_{E,1}}{H_E^*},
\]

because it compares the cosines of the indicated angles on both sides of the inequality without the use of arccosines. To determine whether the satellite is in the Earth's penumbra, it remains to compare the segments \( S'S''' \) and \( S'S \) on line \( \vec{d} \), perpendicular to the line \( \vec{d} \) (Figure 2). From the similarity of the triangles \( \Delta O_H^* S'S''' \) and \( \Delta O_H^* T_{E,1} O_E \) it follows:
\[ S'S'' = \frac{R_E}{\sqrt{H_E^2 - R_E^2}} \cdot \left( H_E^* + r \cdot \frac{\vec{R}_{SE} \cdot \vec{r}_{sat}}{|\vec{R}_{SE}| \cdot |\vec{r}_{sat}|} \right) \]

For the size of the line segment $S'S$ we have again:

\[ S'S = r_{sat} \sqrt{1 - \left( \frac{\vec{R}_{SE} \cdot \vec{r}_{sat}}{|\vec{R}_{SE}| \cdot |\vec{r}_{sat}|} \right)^2} \]

In the penumbra zone, the Sun is partially obscured by the Earth, reducing the flow of light reaching the satellite. This reduction depends on the part of the solar disk covered by the Earth's disk. At low orbits, the time for satellites to pass through the penumbra is short and insignificant compared to the times for passing through the sunlit and umbra zones. However, at higher orbits, this time is longer and could be important to be considered.

**Program realization**

*A situational problem description model*

A descriptor of situational problems is a one-dimensional array whose elements are derived types containing the values of different attributes (parameters and constraints as well as results) of the conditions comprised in the problem. The first (zero) element of the descriptor contains control information and results about the entire situational problem. The following elements contain the values of different attributes (parameters and constraints as well as results) of the situational conditions.

Figure 3 illustrates a general template in Fortran language for the creation of situational problem descriptor objects. The descriptors of different situational problems are combined into a two-dimensional array with dimensions $K \times N$, where $N$ is the number of problems and $K$ is the maximum number of situational conditions among all problems.

- **SitCond** is a derived type that describes different situational conditions. It contains various attributes, which are common for each situational condition. Some of them are: **sit_code** – identification code of situation condition
- **sat_num** - satellite number to which the condition is associated
- **duration** – time interval when the condition is met
- **dt_sit** – local parameter- accumulates duration of condition before ending duration

The **UNION** statement defines groups of fields that share memory among different situational conditions. For the conditions that are discussed here, the following map … end map contains three logical variables **umbra, penumbra** and
sunlit. When the satellite is in a part of the orbit falling in the shadow, penumbra, or illuminated zone, the corresponding variable is assigned a value of true. These variables have control functions. The **UNION** statement defines an array of storage. The **UNION** operator specifies an area of memory that is used polymorphically by each of the functions computing different situational conditions.

```
MODULE
  type   SitCond
    integer  sit_code         ! Code of the situation condition; every situation has identification code
    integer  sat_num          ! Which satellite concern this situational problem
    logical  flag             ! Satisfaction of situational condition: .false. or .true.
    logical  begin_sit        ! If is true – the beginning of situational condition satisfaction
    logical  fl_results       ! If .true. - flag for end of situational conditional’s interval
    real*8   t12(2,3)          ! Determine the last time interval where the condition is satisfied
    real     duration          ! Duration of a current situational condition (event)
    real     dt_sit            ! Local parameter- accumulates duration of condition before ending duration
    real     t_cond_total      ! Local parameter- accumulates total durations for the whole time horizon
  end type
union
  map …    ! Other situational conditions
  end map!
end type
end SitCond

END MODULE
```

**Fig. Derived types for situational problems compilation**
The **sit_problem** is a derived type that describes different situational problems. The UNION statement defines groups of two MAP blocks that describe the elements of one situational problem. The first MAP block describes the zero element of the situational problem descriptor, which contains control parameters and the problem's attributes. The second MAP block allows the inheritance of the properties of each of the situational conditions in the situational problem. The Situational solver interprets situational condition attributes according to the identification code **sit_code**. Each function corresponding to a given situational condition interprets the attributes in a specific way corresponding to a corresponding MAP block.

*Design of the computer subroutines for the umbra/sunlit situational conditions*

Two computer subroutine functions **Sit__78** and **Sit__79** have been developed to check if the satellite is in the shadow or in the sunlight zone respectively (Appendix A). The subroutines are realized in the Fortran language. When carrying out a situational analysis, which is related to the simulation of multi-satellite systems, some tasks refer to observations in the Earth's shadow and others to observations in the sunlit zone. Some calculations about the geometry of the shadow are the same for all satellites. For this reason, the subroutine named **Preliminary_Calculations** is added to increase computational efficiency. It performs these common calculations related to the umbra and light zone, which depend only on the Sun-Earth distance-the height of the conical shadow of the Earth \( H_E \) (expression (3)), analogous quantity \( H_E^* \) and some others. This subroutine, as well as the **Sit__78**, is an additional entry point to the **Sit__79** subroutine. The results of the calculations in **Preliminary_Calculations** are contained in static local variables that are available within the **Sit__79** and **Sit__78** subroutines. When these subroutines are used within the parallel processor for situational analysis (Atanassov, 2016), these variables are common to all computational threads. Some variables (described in operator “automatic”) need to be declared as dynamic to be in the local memory for each thread and the calculations for different situational tasks are not influenced by each other. Besides, the code equal to the two subroutines **Sit__79** and **Sit__78** is differentiated within the internal subroutine **If_flag**. This subroutine has control functions for the situation analysis processor and calculates the values of the variables specific for each situational condition stored in a generic structured variable **SitCond** (Appendix A). The approach allows situational conditions’ subroutines to be reentrant and protected when multi-thread parallelization is applied. **If_Flag** internal function operates with the dummy arguments of **Sit__78** and **Sit__79** functions.

The main purpose of the subroutines **Sit__79** is to check the feasibility of the relevant conditions. The first check through operator **a1:IF** is related to inequality (4). This check expresses the necessary condition to seek a satellite in the
Earth’s umbra. If the necessary condition is fulfilled then distances $S’S''$ and $S’S$ according to the expressions (5) and (6) are calculated. Let us recall that when calculating $S’S’’$, preliminary calculations performed in the subroutine Preliminary_Calculations are used. Checking whether the satellite is in the umbra is performed in the operator a2:IF. If the satellite is not in the umbra, the operator a3:IF checks based on the expressions (6) and (9) whether it is in the penumbra (checking with the a1:IF operator doesn't guarantee either shadow or penumbra).

Subroutine Sit__78 deals with the light zone analogously to Sit__79. The quantity $H_E^*$ (analogous to the height of the Earth’s umbra $H_E$) is presented by expression (7) and calculated in subroutine Preliminary_Calculations as variable H_penu. The a5:IF operator checks the necessary condition (8). The operator a6:IF checks whether the satellite is in the light zone. The operator a7:IF checks based on expressions (5) and (6) whether it is also in the penumbra (regardless of the condition (8)) in case it is not in the light zone.

The determination of the time intervals, when conditions are met, is performed by the internal subroutine If-Flag. The interval begins when the value of the condition changes from false to true. The end of the interval is taken to be the moment when the value of the situational condition changes from true to false.

**Conclusion and outlook**

Situational conditions related to the models of both, the Earth’s shadow and the lit zone are examples of conditions participating in several different situational problems. Such situational problems arise from numerous scientific problems and involve the use of data from diverse instruments located on satellites either of one or more space missions. The attention is directed to the use of these models in the frame of Situational Analysis Solver in the analysis of multi-satellite space missions.

Performing an analysis related to solving a large number of situational problems, each with several conditions can be time-consuming. Apart from the computational models’ optimization of the situational conditions and the application of parallel calculations, the search and research for optimization methods of the situational analysis are an interesting challenge and necessary continuation of the work.

**Acknowledgment**

The author would like to thank Ms. R. Dimitrova for the technical support by preparation of the article.
Appendix A. Source code of the subroutines checking the situational conditions

!*******************************************************************
*
!Sit__78: conic shadow, checks if the satellite is in the sunlit zone, out of umbra
!Sit__79: conic shadow, checks if the satellite is in the penumbra

! Preliminary_Calculations: common for all satellites
! If_Flag: internal subroutine
! Dist_sun – Sun-Earth distance
! H_shad – a height of the conical shadow of the Earth
! Re – radius of the Earth
! Rm_sun – solar radius
! r_sat – modulus of the radius vector of satellite in GeKS

FUNCTION Sit__79 (t,dt,xv,umbra,penumbra,fl_rezults,duration,begin_sit,dt_sit,t12)
USE DFlib
   logical   Sit__79,Sit__78,umbra,penumbra,fl_rezults, begin_sit
   real      duration,
   t12*8(2,3)
   real*8   t,dt,xv(3)
   real*8   r_sat
   real*8   Re/6378.D3/,Rm_Sun/695510.D3/ ! [m] /1.D9/
   real*8   S1S2,S1S3,SpS,SpS
   real*8   H_shad, H_penu, D_shad, D_penu, cos_OhOeTe1, cos_OhOeTe2
   real*8   rS,rSun(3)
   common /cSun_vek/rS,rSun ! Modulus of vector and directional cosines
   logical   flag
   automatic   flag,S1S2,S1S3,S1S,SpSs,SpS,cos_OhOeS
   r_sat       =  SQRT(xv(1)**2 + xv(2)**2 + xv(3)**2) ! |radius - vector| of the satellite - modulus
cos_OhOeS= -(xv(1)*rSun(1) + xv(2)*rSun(2) + xv(3)*rSun(3) )/(r_sat) !- from scalar product

   a1:IF(cos_OhOeS.GT.cos_OhOeTe2) THEN ! It makes sense to check for umbra
   S1S2= D_shad*(H_shad - r_sat*cos_OhOeS )
   S1S = r_sat*SQRT((1.D0 - cos_OhOeS)*(1.D0 + cos_OhOeS));
   a2:IF(S1S.LE.S1S2) THEN ! The satellite is in umbra
   flag=.true.; umbra=.true.
   ELSE ! The satellite is in penumbra eventually
   flag=.false.;
   S1S3= D_penu*(H_penu + r_sat*cos_OhOeS )
   a3:IF(S1S.LT.S1S3) THEN ! Checking for penumbra
   penumbra=.true.;
   ELSE
   penumbra=.false.
   ENDIF a3
   ENDIF a2
   ELSE ! No sense to check in the sunlit zone
   flag=.false.
   ENDIF a1
   CALL If_Flag(Sit__79)
RETURN
ENTRY Sit__78(t,dt,xv,umbra,penumbra,fl_rezults,duration,begin_sit,dt_sit,t12)
   r_sat       =  SQRT(xv(1)**2 + xv(2)**2 + xv(3)**2) ! |radius - vector| of satellite - modulus
cos_OhOeS= -(xv(1)*rSun(1) + xv(2)*rSun(2) + xv(3)*rSun(3) )/(r_sat) !- from scalar product

   a5:IF(-cos_OhOeS.GT.cos_OhOeTe1) THEN ! The satellite is in sunlit zone
   flag=.true.
   ELSE ! No sense to check about umbra and penumbra
   S1S3= D_penu*(H_penu + r_sat*cos_OhOeS )
   S1S = r_sat*SQRT((1.D0 - cos_OhOeS)*(1.D0 + cos_OhOeS))
   a6:IF(S1S.GE.S1S3) THEN ! Satellite is in sunlit zone!!
flag=.true.
ELSE ! the satellite is in penumbra eventually
flag=.false.
S1S2= D_shad*(H_shad - r_sat*cos_OhOeS)
a7:IF(S1S.GT.S1S2) THEN ! Checking for penumbra
penumbra=.true.; umbra=.false.; flag=.false.
ELSE ! outside the penumbra
penumbra=.false.; umbra=.true.; flag=.false.
ENDIF a7;
ENDIF a6
ENDIF a5
CALL If_Flag(Sit__78)
RETURN
ENTRY Preliminary_Calculations()

H_shad= Re*rS/(Rm_Sun - Re); cos_OhOeTe2= Re/H_shad
H_penu= Re*rS/(Rm_Sun + Re); cos_OhOeTe1= Re/H_penu
D_shad= Re/SQRT((H_shad - Re)*(H_shad + Re))
D_penu= Re/SQRT((H_penu - Re)*(H_penu + Re))
RETURN
CONTAINS
SUBROUTINE If_Flag(Sit_cod)
Logical Sit_cod
IF(flag) THEN
IF(.NOT.begin_sit) THEN ! Beginning of the interval
begin_sit=.true.
t12(1,1)= t ! stores the start time
dt_sit=.0; fl_rezults=.false.
ELSE
ENDIF

ELSE ! Satellite isn't visible
IF(begin_sit) THEN ! the situational interval continues
duration= dt_sit - dt; fl_rezults=.true.
dt_sit=.0; begin_sit=.false.
ELSE
duration=.0;
ENDIF
Sit_cod=.false.
ENDIF
END SUBROUTINE If_Flag
END FUNCTION Sit__79

References

РАЗРАБОТКА СИТУАЦИОННЫХ УСЛОВИЙ И ПОДПРОГРАММ ДЛЯ ПАРАЛЛЕЛЬНОГО ВЫЧИСЛИТЕЛЬНОГО ИНСТРУМЕНТА СИТУАЦИОННОГО АНАЛИЗА С ИСПОЛЬЗОВАНИЕМ КОНИЧЕСКОЙ МОДЕЛИ ТЕНИ ЗЕМЛИ

A. Атанасов

Аннотация

Поиск оптимальных временных интервалов выполнения операций спутников основан на проверке конкретных условий геометрического или физического характера. В ситуационной задаче сочетаются несколько условий. Для решения различных ситуационных задач необходимо заранее разработать программные коды для проверки различных ситуационных условий. Представлены два ситуационных условия для определения того, находится ли спутник в освещенной солнцем зоне или в тени на основе конических моделей земной тени. Вводятся необходимые (но не достаточного) условия для находжения спутника соответственно в тени или освещенной солнцем зоне упрощающие расчеты и повышают вычислительную эффективность для большей части орбиты спутника. Задачи с одним или несколькими условиями решаются с помощью разработанного параллельного решаителя для ситуационного анализа. Условия проверки положения спутника относительно тени Земли могут сочетаться с другими ситуационными условиями.