ACCURATE DETERMINATION OF THE BOUNDARIES OF THE INTERVALS IN WHICH SITUATIONAL CONDITIONS ARE FULFILLED. ALGORITHM AND SUBROUTINES

Atanas Marinov Atanassov

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

Keywords: *Space Mission Analysis and Design (SMAD); situational analysis, situational conditions*

Abstract: *Computer simulations are applied at various stages of space mission preparation. Situational analysis is extremely important for assessing/evaluating the possibilities for conducting measurements, based on which a relevant scientific problem can be solved. A computational tool is being developed to solve numerous situational problems. By solving such type of problems check the occurrence of suitable conditions for measurements in the course of the movement of satellites along their orbits. Each situational problem is considered as compiled of one or a conjunction of several situational conditions. Situational conditions are represented by geometric models that include specific parameters and constraints. From a mathematical point of view, they represent predicate functions. Within the framework of simulation models of space missions, situational conditions are checked with a step in the system time, coinciding with the step of numerical integration of the equations of motion of the satellites. When using a larger step, to reduce computational costs, the determination of the boundaries of the time intervals in which the situational conditions are fulfilled may turn out to be too rough. Given the speeds with which a satellite moves in near-Earth space, its location at the beginning and end of an open interval thus determined may differ by hundreds of kilometers. The text of the report presents an algorithm for specifying the boundaries of the interval. The subroutines developed for this purpose, which are applicable to different situational conditions, are shown.*

ТОЧНО ОПРЕДЕЛЯНЕ НА ГРАНИЦИТЕ НА ИНТЕРВАЛИТЕ, В КОИТО СЕ ИЗПЪЛНЯВАТ СИТУАЦИОННИ УСЛОВИЯ. АЛГОРИТЪМ И ПОДПРОГРАМИ

Атанас Маринов Атанасов

Институт за космически изследвания и технологии – Българска академия на науките
e-mail: At_M_Atanassov@yahoo.com

Ключови думи: *анализ и проектиране на космически мисии; ситуационен анализ, ситуационни условия*

Резюме: *Компютърните симулации се прилагат на различни етапи от подготовката на космически мисии. Ситуационният анализ е изключително важен за преценка/оценка на възможностите за провеждане на измервания, на основата на които да може да се реши съответна научна задача. Разработва се изчислителен инструмент за решаване на ситуационни задачи. Чрез такъв тип задачи се проверява възникването на подходящи условия за измервания в ода на движението на спътниците по съответните им орбити. Всяка ситуационна задача се разглежда като съставена от едно или конюнкция от няколко ситуационни условия. Ситуационните условия се представят най-често с геометрични модели включващи параметри и ограничения. От математическа гледна точка, те представляват предикатни функции. В рамките на симулационни модели на космически мисии, ситуационните условия се проверяват със стъпка по системното време, съвпадащо със стъпката на числено интегриране на уравненията на движение на спътниците.При използване на по-голяма стъпка, с цел да се намалят изчислителните разходи, определянето на границите на времевите интервали, в които се изпълняват ситуационните условия, може да се окаже твърде грубо. Предвид скоростите с които един спътник се движи в около-земното пространство, местоположението му в така определените начало и край на открит интервал може да се отличава със стотици километри. В текста на доклада е представен алгоритъм за уточняване на границите на интервала. Показани са разработените за целта подпрограми, които са приложими за различни ситуационни условия.*

Introduction

Computer simulations are performed to clarify various aspects related to the orbital motions of satellites, constructive and functional parameters of space and ground-based subsystems, and payloads, at different stages of space mission preparation. This analysis falls under a separate scientific field, part of space research, known as Space Mission Analysis and Design (**SMAD**) [1]. The Situational Analysis (**SA**) is a crucial component of **SMAD**. It focuses on identifying time intervals or specific moments when conditions are suitable for conducting satellite operations or other orbital events.

A multiphysics programming environment for computer simulations of multi-satellite missions is under development at the Space Research and Technology Institute, BAS. The Parallel Situational Analysis Solver (**PSAS**) is an essential calculation tool within this environment. Mathematical models of diverse situational conditions are being developed, which are necessary for carrying out analyses with the **PSAS**.

State of the problem

Situational analysis studies and establishes the occurrence in the course of orbital movements of satellites of conditions suitable for the performance of satellite operations (measurements, experiments, communications, space debris close approach risk, etc). One situational task may include one or more conditions sc:

(1) $$SP = sc_1 \wedge sc_2 \wedge \ldots \wedge sc_n = \bigwedge_1^n sc_i,$$

The conditions sc_i themselves are predicate functions accepting values of ***truth*** or ***false***. Accurately determining the beginning t_b and end t_e of time intervals when situational conditions are met is an additional but important task of the analysis. Situational condition checks are performed for each time step, after integrating the satellite equations of motion.

At velocities of the order of several kilometres per second at which space objects travel (depending on the altitudes at the points on the orbit), the distances travelled per integration step can be of the order of hundreds of kilometres. Determining the boundaries of time intervals (t_b, t_e) when the situational conditions are met, based on the coordinates of the satellites, determined with a step Δt of the order of tens of seconds and one to two minutes, may be associated with large inaccuracies. Naturally, these boundaries are somewhere within a time interval $(t - \Delta t, t)$ and they should be searched for when events related to the detection of entry and exit from the searched interval occur.

Passage of a satellite through the radio area of a ground station

An approach to determine the boundaries of time slots will be demonstrated based on the requirement that a satellite passes through the radio visibility zone of a ground communication station located at geographical coordinates φ (latitude) and λ (longitude). We will define the station's radio visibility zone by the angle θ^* relative to the zenith (or by the angle δ above the horizon; $\theta^* = .5 * \pi - \delta$ (Figure 1). The condition δ > 0 is imposed to avoid disturbance and information loss.

$$Rz_{station}(1) = Rz * \boldsymbol{SIN}(.5 * \pi - \varphi) * \boldsymbol{COS}(\lambda + \omega * t)$$
$$Rz_{station}(2) = Rz * \boldsymbol{SIN}(.5 * \pi - \varphi) * \boldsymbol{SIN}(\lambda + \omega * t)$$
$$Rz_{station}(3) = Rz * \mathbf{COS}(.5 * \pi - \varphi)$$

In the last formulas ω is angular rotation velocity of the Earth. For the radius vector of the satellite $\vec{R}s_{top}$ in the topocentric coordinate system associated with the observation station, we may write down:

$$\vec{R}s_{top} = \vec{r}_s - \vec{R}z_{station},$$

where $\vec{r}_s$ and $\vec{R}z_{station}$ are the radius vectors of the satellite and the station in the geo-equatorial coordinate system. Then, for the angle Θ between the vectors $\vec{R}s_{top}$ and $\vec{R}z_{station}$ we have:

$$COS(\theta) = \vec{R}s_{top}.\vec{R}z_{top} / \boldsymbol{SQRT}(\vec{R}s_{top}{}^2.\vec{R}z_{station}{}^2)$$

By comparing the cosines of the angles θ and θ^* it is determined whether the satellite is in the radio visibility zone of the ground station.

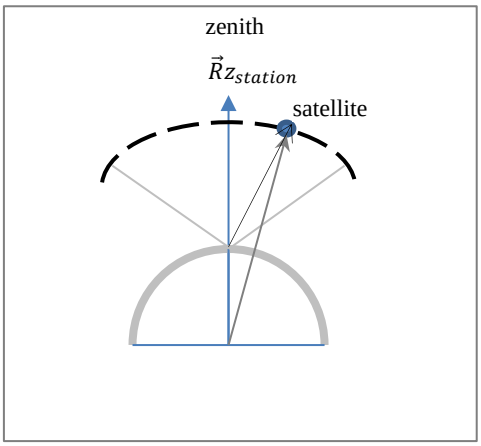

Fig. 1. Geometric model of the situational condition

Interpolation of satellite coordinates

To determine the beginning and end of the time intervals (t_b, t_e) in which the considered situational condition is fulfilled, interpolation of basic quantities included in the model presented above is used. At this stage, the Lagrange method [2] is applied. For a polynomial of n^{th} order:

1. $P_n(x) = \sum_{i=0}^{n} \frac{L(x)}{(x-x_i)L'(x_i)}$

where $L(x) = (x - x_0)(x - x_1) \dots (x - x_n)$ and $L'(x_i) = (x_i - x_0) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n)$. For this purpose, the m last state vectors $\langle\langle \vec{p}(t - (m-1)\Delta t)_1, \vec{p}(t - (m-2)\Delta t)_2, \dots \vec{p}(t)_m \rangle$ of the satellites from one mission are stored. In this case, only the coordinates are used for interpolation, but in others, the velocities are also utilized. After each subsequent integration step of the equations of motion, the obtained values of the components of the state vector $\vec{p}(t)$ are stored as the last element of an array. The values of $L'(x_i)$ are constant and are calculated once at the beginning for all objects. The addresses of these values and the state vectors for the objects from each mission are passed to the situational conditions when creating their descriptors.

The situational condition model involves the geographical coordinates of the station, which, due to the Earth's rotation, are calculated for equidistant moments at which the satellite state vectors are integrated. Therefore, the station coordinates in a geo-equatorial coordinate system also need to be determined by interpolation for moments within a time step.

At the stage of creating the model of a satellite mission (or other main object, for example, space debris), the following structure is created:

```
type      Nodes_Lagrange
  integer    nodes,count_nodes
  integer    denominator_nodes_adr
  integer    t_nodes_adr,
  integer  xv_nodes_adr
end type  Nodes_Lagrange
```

The attributes of this structure are assigned values when the space mission model is created, and its address is written to the mission descriptor. Access to this descriptor is provided when creating situational task models. The address of the mentioned structure with the addresses of the state vectors of the objects, the Lagrange coefficients, and the number of interpolation nodes are taken from this descriptor and passed to the situation solver for solving situational problems. This is done at the creation stage of the mission model after the thread pool is created.

Algorithm and program implementation

PSAS is developed on the basis of the programming language Fortran. The situational condition with which it is checked whether a satellite is in the radio visibility zone of a ground communication station is implemented with the subroutine function **Sit__2**. The subroutine initially collects the necessary data to make it possible to determine the values of variables for moments of

time between the interpolation nodes by interpolation. When this is done, the situational condition is checked for the last moment of time by calling the subroutine function **Radio_zone**. This subroutine implements the situational condition model presented above. The value returned by **Radio_zone**, through the logical variable flag and other variables, is passed to the list of actual arguments of the subroutine **If_Flag_u**. The actual arguments also include the names of subroutines that are used depending on the result of checking the situational condition. The subroutine **If_Flag_u** is designed to be universal, for use in other situational conditions as well. When the situational condition is fulfilled for the first time at some point in time, the subroutine sub1 is called within the **b:IF(.NOT.begin_sit) THEN** construct. The variable sub1 is contextually interpreted as the name of a subroutine declared in **Sit__2** as **Radio_zone_be**. This subroutine is a combination of searching for the beginning and end of the searched interval. The value **.true.** of the first argument specifies the search mode for the beginning. Below, this subroutine is called again with the value .false. for the search mode for the end of the interval. The last argument **sub3**, in the list of the subroutine **sub1** (or **Radio_zone_be**), also contains the name of a subroutine - **Radio_zone**. The last subroutine is called within the **Radio_zone_be** subroutine. In the **Radio_zone_be** subroutine, the **Lagr_interp** subroutine is called to interpolate the geographical coordinates of the station, transformed into **GECS**, and the **Lagr_interp1** subroutine to interpolate the satellite coordinates. The differences in these subroutines are minimal – the first interpolates one value, and the second three. In the subroutine **Radio_zone_be**, an iterative process is performed in which checks are made of the situational condition in the middle of the interval. Depending on the result of the check, the interval is halved and the condition is checked with the subroutine **Radio_zone**. The process continues until the size of the interval reaches a value specified by **Tool**. The iterative process is fast enough and is terminated after about ten iterations.

The complex of subroutines for checking the considered situational condition is intended to function within the PSAS. To avoid problems in parallel calculations, local variables from different threads must be protected from mutual influence. This can happen when solving the same type of situational tasks from different threads simultaneously. This is avoided by declaring local variables as automatic.

Application of the proposed algorithms

To illustrate the proposed algorithms, we will consider the motion of a satellite with initial orbital parameters – semi-major axis $a = 7{,}200{,}000$ km, eccentricity $e = .001$ and inclination $i = 5°$. We will use the coordinates of the ground station ($\lambda = 10°, \varphi = 5°$) and the angle above the horizon $\theta = 10°$. The numerical integration of the satellite motion equation was performed with a step of 20 sec.

Table 1.

	Input time by interpolation [s]	Without interpolation		Output time by Interpolation [s]	Without interpolation	Duration [s]
4241.0	4259.4140	4261.0	4861.0	4867.9760	4881.0	608.562
10801.0	10805.3216	10821.0	11401.0	11420.4791	11421.0	615.157

Conclusion and future work

The proposed approach for accurately determining the start and end points of time intervals during which a specified situational condition is satisfied can be extended to other types of situational conditions. Research in this direction is ongoing, with the development of an alternative methodology currently underway. Furthermore, the Lagrange interpolation method employed in this study may be compared with other interpolation techniques to elucidate their respective advantages and limitations. In the present case, the interpolation time interval is constrained between the final two interpolation nodes - the penultimate and the last moment of time. Investigations into interpolation based on non-polynomial functions, as well as approximation methods, would also be beneficial, particularly in the context of advancing the multiphysics simulation environment for space mission analysis, within which the situational analysis tools are being developed.

The methodology proposed for the precise identification of the onset and termination of time intervals during which a given situational condition is satisfied can be generalized to accommodate other types of situational constraints. Ongoing research in this area includes the development of an alternative computational framework aimed at enhancing robustness and applicability. Additionally, the Lagrange interpolation technique utilized in the present study may be systematically compared with other interpolation schemes to assess their relative strengths and limitations in the context of

situational analysis. In this specific implementation, the interpolation interval is bounded by the final two interpolation nodes—namely, the penultimate and terminal epochs—serving as a constraint in the numerical procedure. Further exploration of interpolation strategies based on non-polynomial basis functions, as well as approximation techniques, would be of significant value, particularly in light of the continued development of a multiphysics simulation environment for space mission modeling, within which the situational analysis tools are being integrated.

References:

1. Wertz, J.R., Larson, W.J., Space Mission Analysis and Design. Microcosm Press, Kluwer Academic Publishers, third ed., 1999
2. Atanassov, A.M.. Parallel satellite orbital situational problems solver for space missions design and control. *Advances in Space Research*, *58* (9), 2016, pp. 1819–1826
3. Atanassov, A.M., Applying Parallel Situational Analysis Solver to Satellite–Space Debris Close Approach Problems. Algorithms and Subroutines. *Aerospace Research in Bulgaria*, v *37*, 2025, pp. 16–29
4. Кальницкий, Л.А., Добротин, Д.А., Жевержеев, В.Ф., and Сапогов, Н.А., Специальный курс высшей математики для втузов, 1976.

APPENDIX A

```
!*****************************************************************************************************************************
! Sit2: Determines whether the satellite is visible from a ground station with coordinates (lati_sta,long_sta) and
!                    (lat0,long0)
!          begin_sit- sit.cond. satisfaction beginning =>.true.  stored in t12(1)
!          begin_sit- sit.cond. satisfaction endind      =>.false. stored in t12(2)
!          dt_sit= t12(2) - t12(1); duration= dt_sit
!          begin_sit=.false => fl_rezults=.true. - the results are ready- for one step only!
!
!          address- where are stored <lati,long> coord
!.....*
FUNCTION        Sit__2(t,dt,object1_adr,nodes,nodes_count,node_t_adr,adr_znam_nodes,address, &
                       lati_sat,long_sat,lati_sta,long_sta,angle,fl_rezults,duration,begin_sit,dt_sit,t12)
 USE  DFlib
 external                    Radio_zone_be
 logical            Sit__2,   fl_rezults,begin_sit
 integer             object1_adr,              node_t_adr,adr_znam_nodes,address
 real                lati_sat,long_sat,lati_sta,long_sta,              duration
 real                lati_long(2,nodes),           t12*8(2,3)
 real*8                 t,dt,xv(6)
!_____________________________________________________
 type     sit_cond_2
  integer     object1_adr,node_t_adr,adr_znam_nodes,address
  real       lati_sat,long_sat,lati_sta,long_sta,angle
 end type  sit_cond_2
 type     (sit_cond_2) cond_2,      cond_2_lenght(2)
  integer           cond_2_addr, cond_2_len
!_____________________________________________________
 logical     Radio_zone
 real        dt_sit
 real*8              object1_nodes(6,nodes),node_t(nodes),node_znam(nodes)
 real*8    Rz/6371.D3/,pi/3.141592654/,grrad
 logical     flag, flag1,flag2
AUTOMATIC

  POINTER(node_t_adr,node_t); POINTER(object1_adr,object1_nodes);
  POINTER(addressll,lati_long);  POINTER(address,icounter);
  POINTER(cond_2_addr,cond_2)
        cond_2_addr= address + 2*4*nodes+4+4; addressll= address + 8

  IF(icounter.LT.nodes) THEN
     icounter= icounter + 1;
         lati_long(1,icounter)= lati_sat;      Sit__2=.false.
         lati_long(2,icounter)= long_sat));  RETURN
     cond_2_len      = LOC(cond_2_lenght(2)) - LOC(cond_2_lenght(1));  RETURN
                           ELSE; addressll= address + 8
          lati_long(1,1:nodes-1)= lati_long(1,2:nodes)
          lati_long(1,   nodes   )= lati_sat
          lati_long(2,1:nodes-1)= lati_long(2,2:nodes)
```

```
          lati_long(2,   nodes   )= long_sat
     ENDIF

cond_2%adr_znam_nodes= adr_znam_nodes
cond_2%object1_adr      = object1_adr;      cond_2%node_t_adr= node_t_adr;
cond_2%lati_sat         = lati_sat;         cond_2%long_sat  = long_sat;
cond_2%lati_sta         = lati_sta;         cond_2%long_sta  = long_sta;
cond_2%angle            = angle;            cond_2%address   = address + 8

    flag= Radio_zone(node_t(nodes),dt,object1_nodes(:,nodes),lati_sat,long_sat,lati_sta,long_sta,angle,0);

   CALL  If_Flag_u(Sit__2,flag,node_t(nodes),dt,fl_rezults,duration,begin_sit,dt_sit,t12, &
                           Radio_zone_be,Radio_zone,nodes,LOC(cond_2))

                    Sit__2= flag
END FUNCTION  Sit__2

FUNCTION          Radio_zone(t,dt,xv,lati_sat,long_sat,lati_sta,long_sta,angle,kod)
  USE  DFlib
  logical         Radio_zone
  real                 lati_sat,long_sat,lati_sta,long_sta,angle
  real*8          t,dt,xv(6)
!_____________________________________________________
  real*8     Rz_top(3),Rs_top(3),cos_tita
  real*8     r,r2,a,b,c,s1,q,l,sin_a,sin_b,cos_a,cos_b,Tita ! a/1.D0/
  real*8    Rz/6371.D3/,pi/3.141592654/,grrad
  logical    flag
!_____________________________________________________
   real*8          tq,TU,dtq,duration,omEarth/.729211575D-4/
   common    /c_time/tq,TU,dtq,duration
!_____________________________________________________
  AUTOMATIC   r,r2,c,s1,l,q,Tita,a,b,sin_a,sin_b,cos_a,cos_b,cos_c

     Rz_top(3)= Rz*COS(.5*pi - lati_sta)
     Rz_top(1)= Rz*SIN  (.5*pi - lati_sta)*COS(long_sta+TU*omEarth)
     Rz_top(2)= Rz*SIN  (.5*pi - lati_sta)*SIN (long_sta+TU*omEarth)
     Rs_top(1)= xv(1) - Rz_top(1); Rs_top(2)= xv(2) - Rz_top(2); Rs_top(3)= xv(3) - Rz_top(3)
    !Rs_top    = xv - Rz_top
     cos_tita   = (Rs_top(1)*Rz_top(1) + Rs_top(2)*Rz_top(2) + Rs_top(3)*Rz_top(3))/ &
          SQRT((Rs_top(1)**2 + Rs_top(2)**2 + Rs_top(3)**2)*(Rz_top(1)**2 + Rz_top(2)**2 + Rz_top(3)**2))
  IF(COS(angle).LE.cos_tita) THEN
                    flag=.true.
                      ELSE
                    flag=.false.
    ENDIF

           Radio_zone= flag
END FUNCTION  Radio_zone

!**************************************************************************
! <Radio_zone_begin>- Precizing in/out radio-zone moments
!
!            flag - .true. for 'in' & .false. for 'out'
!          mode - .true. for begin & .false. for end of time interval
!.....................................................................................*
SUBROUTINE         Radio_zone_be(mode,nodes,t_begin,adr_cond,Radio_zone)
  USE    DFlib
    logical              Radio_zone, mode
    integer                          adr_cond
    real*8                      t_begin
    real*8       object1_nodes(6,nodes),node_t(nodes),node_znam(nodes)
!_____________________________________________________
    type      sit_cond_2
     integer     object1_adr,node_t_adr,adr_znam_nodes,address
     real        lati_sat,long_sat,lati_sta,long_sta,angle
    end type  sit_cond_2
    type     (sit_cond_2) cond_2
!_____________________________________________________
  logical   flag,log,flag1,flag2,flag3
```

```
 integer    object1_adr,adr_znam_nodes,node_t_adr
 real       lati_sat4,long_sat4
 real       lati_sat,long_sat,lati_sta,long_sta,angle,lati_long(2,nodes),latitude(nodes),longitude(nodes)
 real*8     lati_sat8,long_sat8
 real*8    Rz/6371.D3/,pi/3.141592654/,grrad
 real*8   dt,tool/.001/,td,tg,tm
 real*8   xv(6),t,tt
  POINTER(adr_cond,cond_2)
  POINTER(node_t_adr,node_t);             POINTER(adr_znam_nodes,node_znam)
  POINTER(object1_adr,object1_nodes); POINTER(address,lati_long)

      object1_adr     = cond_2%object1_adr;       node_t_adr = cond_2%node_t_adr;
   adr_znam_nodes= cond_2%adr_znam_nodes;
               lati_sat= cond_2%lati_sat;                long_sat = cond_2%long_sat;
               lati_sta= cond_2%lati_sta;                long_sta = cond_2%long_sta;
                  angle= cond_2%angle;                    address = cond_2%address

      td= node_t(nodes-1); tg= node_t(nodes);;
      dt= tg - td; tm= td + .5D0*dt

  latitude= lati_long(1,:);longitude= lati_long(2,:);

DO WHILE(dt.GE.Tool);
      log= Lagr_interp  (tm,nodes,node_t, latitude,    node_znam,   lati_sat8);   lati_sat4= lati_sat8
      log= Lagr_interp  (tm,nodes,node_t, longitude, node_znam,long_sat8); long_sat4= long_sat8;
      log= Lagr_interp1(tm,nodes,node_t, object1_nodes, node_znam,xv);
     flag= Radio_zone  (tm,dt,xv,     lati_sat4 ,     long_sat4 ,lati_sta,long_sta,angle,1);
  IF(flag) THEN;
       IF(mode) THEN; tg= tm;
                ELSE; td= tm
          ENDIF
            ELSE
       IF(mode) THEN; td= tm;
                ELSE; tg= tm
          ENDIF
    ENDIF
    tm= td + .5D0*(tg - td); dt= tg - td;
END DO

      t_begin= tm;

END SUBROUTINE  Radio_zone_be
!*************************************************************************************************
! <Lagr_interp>- one dimensional interpolation
!
!
!     Sx,Sy,Sz- x,y,z interpolated coordinates
!.....................................................................................*
logical FUNCTION  Lagr_interp (t,nodes,node_t,node_s,node_znam,xv)
  real*4                                                 node_s (nodes)
  real*8                                t,          node_t(nodes), node_znam(nodes)
  real*8                                                                                xv
  real*8                     rt
  real*8             numerator,Sx, coeff_Lagrange
  AUTOMATIC  numerator,Sx, coeff_Lagrange,rt

          Sx= 0.D0;  rt= t;
   DO  nd=1,nodes
        numerator = 1.D0;
       DO md=1,nodes
          IF(md.NE.nd) THEN;
             numerator = numerator *(rt - node_t(md))
            ENDIF
       END DO;
    coeff_Lagrange= numerator *node_znam(nd);
                 Sx= Sx + node_s(nd)*coeff_Lagrange
  END DO;        xv= Sx;

                       Lagr_interp =.true.
```

```
END FUNCTION      Lagr_interp

!*****************************************************************************************
! <If_Flag_u>- for all sit conditions with precizing begin and final times
!.....................................................................................*
SUBROUTINE      If_Flag_u(Sit__cod,flag,t,dt,fl_rezults,duration,begin_sit,dt_sit,t12, &
                                                sub1,sub3,nodes,adr_cond)
 USE  DFlib
   logical                      Sit__cod,  flag,     fl_rezults,         begin_sit
   integer                        adr_cond
   real                                    duration,               t12*8(2,3)
   real*8                       t,dt,H
!_____________________________________________________________________________________
  real*8             t_begin,t_final

a:IF(flag) THEN
  b:IF(.NOT.begin_sit) THEN ! Nachalo za situacijata- promjana na 'begin_sit' kam 'true'
        CALL  sub1(.true.,nodes,t_begin,adr_cond,sub3)
              begin_sit  =.true.;   ! fl_rezults=.false.
                    t12(1,1)= t_begin  ! Remembering the start time
                      dt_sit  =.0; fl_rezults=.false.
                        ELSE ! Inside the situational interval
                    t12(2,1)= t            ! Remembering time with a fulfilled condition
      ENDIF b
          dt_sit= dt_sit + dt
       Sit__cod=.true.
           ELSE ! The condition isn't meet
      c:IF(begin_sit) THEN      ! So far, there has been a situation
            CALL  sub1(.false.,nodes,t_final,adr_cond,sub3)
                    t12(2,1)= t_final   ! Remembering the end time
                    duration= dt_sit-dt; fl_rezults=.true.;!stop
                       dt_sit=.0; begin_sit=.false.
                ELSE
                     duration=.0;
          ENDIF c
         Sit__cod=.false.
    ENDIF a

END SUBROUTINE  If_Flag_u
```