APPLYING SITUATION ANALYSIS SOLVER TO SATELLITE-SPACE DEBRIS CLOSE APPROACHES PROBLEMS. INTERFACE BETWEEN 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 solver, orbital event, situational condition, satellite-space debris close approaches, conjunction analysis.*

Abstract: *As the number of active and passive satellites of the Earth increases, the probability of a direct collision either with some other satellite or with space debris grows. Analysis of the functionality of space systems over time must take into account the possibility of such events, as well as the maneuvers performed to mitigate the danger of a direct collision. The possibility of using the developed processor for situational analysis to solve close approach problems between multi-satellite systems and space debris is considered. This processor is one of the developed computing tools at the core of a multiphysics environment for space mission simulations. The focus is on the interface between models of multi-satellite systems and space debris systems.*

ИЗПОЛЗВАНЕ НА ПРОЦЕСОРА ЗА СИТУАЦИОНЕН АНАЛИЗ ПРИ РЕШАВАНЕ НА ЗАДАЧИ ОТ ТИПА „СБЛИЖАВАНЕ МЕЖДУ СПЪТНИК И КОСМИЧЕСКИ ОТЛОМКИ“

Атанас 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, процесор за ситуационен анализ, орбитални събития, ситуационни условия, сближаване между спътници и космически отломки;.*

Резюме: *С нарастване на броя на активните и пасивни спътници на Земята се увеличава вероятността от пряко стълкновение, както между спътници, така и между спътници и космически отломки. Анализът на функционалността на космическите системи във времето трябва да отчита възможността от такива събития, както и от извършваните маневри за смекчаване на опасността от пряко стълкновение. Разглежда се възможността за използване на разработения процесор за ситуационен анализ за решаване на задачи на сближаване между много-спътникови системи и космически отломки. Този процесор е едно от изчислителните средства в основата на мултифизична среда за симулации на космически мисии. В центъра на вниманието е интерфейса между модели на много-спътникови системи и системи от космически отломки.*

Introduction

In recent decades, human activity in space has been increasing and this is expected to continue in the future [1]. Missions involving different numbers of satellites of different classes are developed and implemented. Federations of missions aimed at solving common tasks are considered. Horstmann at all. [2] consider various sources of the increase in the number of space debris. Lewis and Marsh [3] discuss various patterns of debris growth. The increasing number of space debris is emerging as a significant problem in the near future.

The development of models for computer simulations of the functioning of satellite systems in the conditions of the hostile environment space environment (in particular, the impact of space debris) can provide helpful information at various stages of the preparation to the realization of the missions. The inclusion in the analysis of satellite missions and space debris reveals opportunities for solving

tasks related to reducing the danger of direct collisions, as well as possible collisions [4]. The modification of the functionality of multi-satellite missions/systems to solve relevant tasks can also be explored further.

Algorithms, methods, and tools for simulations of space missions are developed at IKIT. This paper presents part of the work on the development of a situational processor for determining close approaches between satellites and space debris. Attention is focused on the semantics of communications between models within a multiphysics environment for space mission simulations.

Development of tools for Multiphysics simulation of satellite missions

In [6], a multiphysics multi-layer simulation model including various basic objects was presented. Fig. 1 shows a part of this model that is relevant to the present consideration.

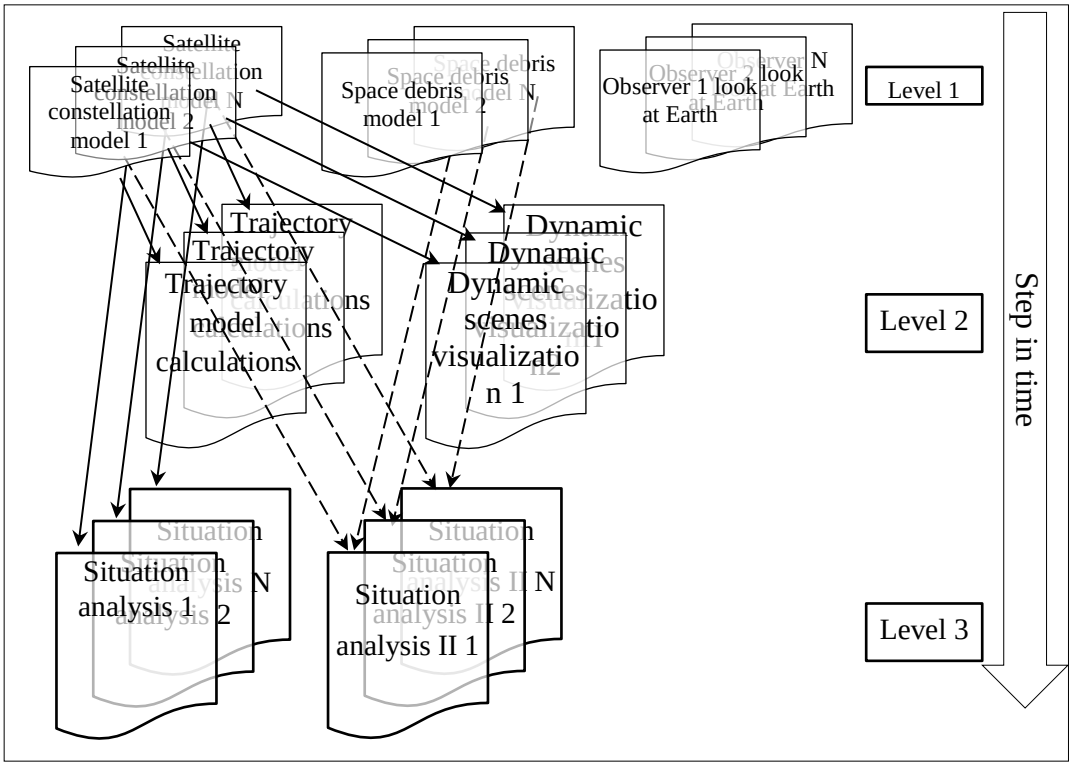

Fig. 1. Part of the conceptual scheme of multiphysics environment for space mission simulation

The different models are located at levels corresponding to the sequence of execution of the respective solvers. At the first level are models of multi-satellite systems and space debris. Other models at this level are "Observer" models, which are auxiliary in nature and used to display dynamic scenes. At the second level are models related to the calculation of parameters of the space environment at points of the trajectory of the satellites from the various satellite systems included in the general simulation model. At this level are models related to the visualization of some dynamic scenes. At the third level are models of situational analysis.

The presented collections of models, which are vertically related to models of basic space objects (satellites, systems of satellites, and space debris, all in the conditions of the atmosphere, ionosphere, and magnetosphere) (can) be connected to each other through interfaces described in corresponding descriptors [6]. Based on these descriptors, relevant solvers take the results of the previous ones and thus simulate various processes (mechanical, situational, mechatronic, and functional related to the operation of subsystems and payload of individual satellites).

In the conditions of federations between separate satellite systems, connections between some models from different systems connected to major space objects are also necessary. Examples of such connections are tasks for convergences between active satellites and objects from different populations of passive, failed satellites, debris from satellites, and other space debris.

Each model is characterized by specific attributes. Setting attribute values to a model defines a corresponding object. The relationships of an object to other objects are also important. The attributes of individual objects are stored in descriptors. Each level of Figure 1 has a separate descriptor.

The number of descriptors for a level and the starting address at which it is located are shared by a named common area. For this purpose, subroutine **add_object** is used (Fig. 2) [7].

```
external      SatelliteIntegratorUPC
  …
CALL  CreatePoolThreads(SatelliteIntegratorUPC,NumThreads,Sit_threads_par,Sit_ha)

  AI_pool_par%AI%num_threads    = NumThreads
  AI_pool_par%AI%    ha_race       = AI_handler_addr  !ha_1                    a)
  AI_pool_par%AI%counter_adr      = AI_GlbCount_addr)
  AI_pool_par%AI%thread_par_adr= PoolParam_addr
  AI_pool_par%AI%granulation       = Granularity

   CALL  add_object(num_AIs,AIs_descriptor_adr,AIs_descriptor_adr,AI_pool_par)
!_____________________________________________________________________________
  IVP_par_SpCon%kind              = 1
  IVP_par_SpCon%sort              = 2
  IVP_par_SpCon%name              = s_const_name                    b)
  IVP_par_SpCon%IVP%solver_index    = num_AIs;
  IVP_par_SpCon%IVP%num_objects    = Numsat
  IVP_par_SpCon%IVP%t_adr          = LOC(t); IVP_par_SpCon%IVP%dt_adr   = LOC(dt);
  IVP_par_SpCon%IVP%xvn_adr       = xvn_adr;
  IVP_par_SpCon%IVP%xvk_adr       = xvk_adr;
  IVP_par_SpCon%IVP%eps_adr       = eps_adr;
  IVP_par_SpCon%IVP%adr_Grv_model = adr_perturbations;
  IVP_par_SpCon%IVP%len_Grv_model = len_Grv_model
  IVP_par_SpCon%IVP%transfer_data_adr= transfer_adr
  IVP_par_SpCon%IVP%work_data_adr  = work_adr
```

Fig. 2. a). Descriptor creation of solver descriptor; б). descriptor creation of initial values problem

Figure 2.a shows a descriptor of computing processors. The attributes it includes are: **num_threads**- number of threads, **thread_par_adr**- memory address where the solver thread pool parameters are saved, **ha_race**- memory address where an event handler is stored, which is used for synchronization between solver threads, **counter_adr**- subtask counter address and **granulation**- number of subtasks solved at once by one thread. With the **add_object** subroutine, the descriptor is written in memory together with other descriptors under a number, which is written in a special **solver_index** attribute of the satellite constellation descriptor (figure 2.b). This second descriptor contains various attributes that fully define the satellite constellation model. Further development of the model may lead to more attributes. The **add_object** routine has polymorphic properties and can be used for different descriptors.

Situational problem composer

An editor representing a dialogue form with various controls has been developed for composing situational problems. First, a space mission is selected among those displayed in the list box control. Depending on the type of the multi-satellite satellite system, they are selected from among possible (contextually determined) situational conditions and displayed in another list box control. The remaining objects are displayed in another window. One of them can be specified for the composition of situational tasks between two multidimensional objects (a close approach between satellites and space debris). Situational tasks between satellites from different multi-satellite missions within a federation are possible. Information is then taken from the descriptors of the selected objects. Some of this information relates to the dimensionality of the objects (number of satellites and number of space debris). Other information about the addresses in the memory where the state vectors of the satellites and space debris are located is copied also from descriptors of the space mission and space debris. The two objects (space mission and space debris population) must be created in advance - one based on a model for a multi-satellite system and the second, a collection of space debris (figure 1).

After collecting the necessary parameters and compiling situational tasks, an actual situational solver is created to solve the compiled tasks. The descriptor of this solver is added to the already created descriptors with the following subroutine (Figure 3.a).

In addition, a descriptor of situational tasks is created (Figure 3.b). This handle is used to communicate with other solvers as well as to control the situation solver. Information about the solver is recorded in the descriptor of the situation problems. Using this information, the solver is started for each time step.

Access to the addresses of the space debris state vectors is provided through a named common area **/cDebris/**.

Situational problem solver

A situational processor was developed for solving high-dimensional problems [5]. For this purpose, it is parallelized based on computational flows (thread parallelization). Each actual situational processor is created by a special subroutine with the launch of a specified number of computational threads to solve a separate group of situational tasks. Different groups of situational tasks can be solved within separate situational models. The information required for processing by the situational solvers is passed to them from parent computing threads through buffer subroutines [5]. One part of this information is contained in situational task descriptors (Figure 4). The management of the computations by the threads of the situation processor is done by a universal subroutine for managing different solvers [8]. This subroutine calls the real subroutine for situational analysis (the actual, specific part of the processor). It in turn calls separate subroutine functions, each describing a predicate function representing a separate situational condition.

```
 external                    SituationProcessorInterUPC                                          a).
CALL  CreatePoolThreads(SituationProcessorInterUPC,NumThreads,Sit_threads_par,Sit_ha)
_____________________________________________________________________________________
  StPrb_param%kind            = 1      ! kind of the solver                                    b).
  StPrb_param%name            = TRIM(IVP_par(kod_IVP)%name)//"/SitTask"
  StPrb_param%SitProb%pool_index = num_AIs
  StPrb_param%SitProb%IVPs_index = kod_IVP            ! num_IVPs
  StPrb_param%SitProb%num_objects= num_sat            ! number of satellites
.. StPrb_param%SitProb%max_num_sit= max_prob_cond  ! max number of sit conditions
  StPrb_param%SitProb%num_sci_task= current_prob      ! number of sit problems
  StPrb_param%SitProb%addr_sit_prob= adr_sit_probs    ! address in memory
.. StPrb_param%SitProb%TrParam_adr  = TrajectoryParam_addr
_____________________________________________________________________________________
CALL  add_object(num_StPrs,StPrb_descriptor_adr    =StPrs_descriptor_adr, &                  c).
                  StPrb_descriptor_adr_new=StPrs_descriptor_adr,   level__3=StPrb_param);
```

Fig. 3. a). Creation of situational solver; b). preparation of the situational descriptor; c). Creation of situational descriptor

```
                          SUBROUTINE      SituationProcessorInterUPC(th_id_num)
 external      Psitanal_UPC
…
  integer            xvn_debris_adr,xvk_debris_adr;          logical         inter;
  common   /cDebris/inter,xvn_debris_adr,xvk_debris_adr
…
 task_descriptor%num_sat          = num_sat
 task_descriptor%t_adr            = t_adr;              task_descriptor% dt_adr = dt_adr
 task_descriptor%xvn_adr          = xvn_adr;            task_descriptor%xvk_adr= xvk_adr
 task_descriptor%max_num_sit    = max_num_sit
 task_descriptor%num_sit_prob    = num_sit_prob
 task_descriptor%sci_problem_adr= sci_problem_adr
 task_descriptor%len_sci_task      = sci_task_len
 task_descriptor%TrajectParam_adr= TrajectParam_adr
 task_descriptor%TrajectParam_len= TrajectParam_len;
 task_descriptor%inter              = inter
 task_descriptor%xvn_debris_adr  = xvn_debris_adr;
 task_descriptor%xvk_debris_adr  = xvk_debris_adr;          local_task_descriptor_adr= LOC(task_descriptor)

 CALL  UPC(th_id_num,num_Sit_threads,ha_1,adr_glb_counter,thread_par_local,granule, &
              Psitanal_UPC,local_task_descriptor_adr,local_num_sit_prob)

END SUBROUTINE  SituationProcessorInterUPC
```

Fig. 4. Part of the buffer subroutine illustrates the transmission of the necessary data to the situation solver via the derived type variable **task_descriptor**. The presence of an external relationship between objects is determined by the value of the variable named **inter**.

To ensure the possibility of checking situational conditions whose models require information from different basic objects, it is first necessary to have access to such information. The situational analysis processor obtains the starting addresses of this information for each object system. For each individual pair of "satellite-debris" or "satellite-satellite" objects, a separate situational task is compiled. Access to the data for each individual object is achieved by indexing relative to the initial address.

```
type      SitCond
  …  ! General attributes
  UNION
  …
   MAP
     UNION
      MAP
        integer    id_debris   ! satellite - space debris
      END MAP
      MAP
        integer    num_sat_61   ! satellite - satellite
      END MAP
     END UNION
     real    distance
     integer   interpolation_nodes ! Lagrange
                                   ! interpolation
     integer   node                ! Caunter
     real*8    node_t (5)          !
     real*8    nodes(6,5,2)  ! Storage for
                             ! interpolation
   END MAP
  END UNION
end type  SitCond
```

Fig. 5. Part of the situational condition model for close approach represented by a derived type variable. There is a separate attribute (**id_debris** or **num_sat_61**) for each variant ("satellite - space debris" or satellite - satellite)

"Close approach" situational condition

Each situational task involves one or more situational conditions. For situational tasks related to the assessment of close approaches between satellites and space debris, one such condition is sufficient. This condition is related to checking if the distance between two objects $D(t)_{sat,debris;i}$ (satellite-space debris) is within a specified interval $D_{threshold}$:

1. $$D(t)_{sat,debris;i} = \sqrt{(x(t)_i^s - x(t)_i^d)^2 + (y(t)_i^s - y(t)_i^d)^2 + (z(t)_i^s - z(t)_i^d)^2} < D_{threshold}$$

In this expression, the index i is the identification number of an individual fragment from a whole population. The coordinates of satellites and space debris are obtained based on numerical integration with a constant step, for discrete moments of time, by models and solvers shown in Figure 1 at the first level. For more efficient calculations, the integration step can be within tens of seconds to a minute or two. In such a case, space objects travel relatively large distances, and time intervals in which $D_{sat,debris;i} < Distance$ (set parameter) cannot be determined based solely on the coordinates calculated with numerical integration. These intervals are easily skipped, especially when the orbital planes of the objects subtend large angles. This problem is solved by interpolating at intermediate points within an integration step ∆t. Various interpolation methods can be used for this purpose [9]. In the present work as a start, applying the Lagrange method is reported. The coefficients are calculated once for the satellite and the other object at each time step.

In addition to calculating the minimum distance between the two objects, the relative velocity $v(t)_{s,d;i}^{rel}$ at the point of conjunction is also calculated:

2. $$v(t)_{s,d;i}^{rel} = \left| \sqrt{\left(v_{x,i}^s - v_{x,i}^d\right)^2 + \left(v_{y,i}^s - v_{y,i}^d\right)^2 + \left(v_{z,i}^s - v_{z,i}^d\right)^2} \right|.$$

Calculating the magnitude of this velocity is important in the event of an impact. This speed is calculated in case the distance between the objects is within specified limits $D_{sat,debris;i} < Distance$. The angle θ_i between the two vectors $\vec{v}_i^s$ and $\vec{v}_i^d$ is also relevant:

3. $$cos\theta_i = \frac{v_{x,i}^s.v_{x,i}^d + v_{y,i}^s.v_{y,i}^d + v_{z,i}^s.v_{z,i}^d}{\sqrt{\left(v_{x,i}^s\right)^2 + \left(v_{y,i}^s\right)^2 + \left(v_{z,i}^s\right)^2}.\sqrt{\left(v_{x,i}^d\right)^2 + \left(v_{y,i}^d\right)^2 + \left(v_{z,i}^d\right)^2}}.$$

A detailed description of the algorithms and programs for searching the time interval where (1) is fulfilled and determining the minimum distance between objects (within this interval) will be presented in another work.

Conclusion

The editor for composing situational tasks and the processor for situational analysis have been further developed to achieve a change in the semantics of the relationships between models related to different underlying objects. We are also working on a situational condition for checking the conjunction between active satellites and space debris below a certain threshold of admissibility. After the initial development of the main additional means, optimization is forthcoming.

Essential in the work is the interface between models describing a sequence of processes related to different multidimensional objects (space debris, multi-satellite systems). This approach can be used to simulate communication between satellites from different satellite systems within a federation. For this purpose, changes have been made to both the modeling tools (models of situational tasks) and the processor for solving situational tasks.

Reference:

1. Selva, D., A. Golkar, O. Korobova, I. L. I. Cruz, P. Collopy et al. (2017) Distributed earth satellite systems: What is needed to move forward?, Journal of Aerospace Information Systems, 14(8), 412–438.
2. Horstmann, A., Kebschull, C., Müller, S., Gamper, E., Hesselbach, S., Soggeberg, K., Ben Larbi, M.K., Becker, M., Lorenz, J., Wiedemann, C. and Stoll, E., 2018. Survey of the current activities in the field of modeling the space debris environment at TU Braunschweig. *Aerospace*, *5*(2), p.37.
3. Lewis, H. G. and Marsh, N., 2021, May. Deep time analysis of space debris and space sustainability. In *Proc. 8th European Conference on Space Debris*.
4. Smirnov, N. N. ed., 2001. *Space Debris: Hazard Evaluation and Debris*. CRC Press.
5. Atanassov, A. M., 2016. Parallel satellite orbital situational problems solver for space missions design and control. *Advances in Space Research*, *58*(9), pp. 1819–1826.
6. Atanassov, A. and Atanassova, L., 2020. Development of Tools for Models' Design of Systems of Multi-Satellite Systems. *Proceedings SES*, *2020*, pp. 110–115.
7. Atanassov, A. M., 2015. Development Classes of Objects' Descriptors for Space Missions Simulation. *Proceedings SES*, *2016*, pp. 51–55.
8. Atanassov, A., UNIFICATION OF "POOL OF THREADS" CONTROL IN THE FRAMES OF DIFFERENT PARALLEL SOLVERS. In Proceedings of SES 2016, pp. 42–46.
9. Кальницкий, Л. А., Добротин, Д. А., Жевержеев, В. Ф., and Сапогов, Н. А., 1976. Специальный курс высшей математики для втузов.