MULTIPHYSICS MODELLING AND SIMULATION FRAMEWORK FOR SATELLITE MISSIONS: CURRENT STATUS AND FUTURE PROSPECTS

Atanas Atanassov

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

Keywords: *Space Mission Analysis and Design, multi-satellite mission, space mission federation,* system of heterogeneous multi-satellite *missions*, *functional models of scientific instruments*

Abstract: *The space sector is rapidly evolving from single, massive satellites to multi-satellite systems. This architectural shift presents itself in two primary forms: satellite constellations, which are homogeneous (or heterogeneous) fleets comprising satellites with varying instruments, masses, dimensions, power requirements, and information and communication needs under a single operator; and Federated Satellite Systems (FSS), which share resources to achieve collaborative goals.*

Designing and managing these highly complex, orbital systems makes advanced computer modelling and simulation indispensable. Traditional design paradigms are no longer sufficient to handle the scale, tight subsystem coupling, and real-time operational constraints. Looking forward, the next frontier in aerospace engineering relies heavily on the development of unified, interdisciplinary simulation frameworks. By bridging diverse domains – such as orbital mechanics, thermal dynamics, telecommunications, and power systems – these frameworks enable comprehensive space mission validation. Simultaneously, the massive scale of distributed space systems generates exponential computational challenges at different stages of mission preparation. Resolving these bottlenecks requires highly efficient, parallel algorithms.

Within the framework of the space mission modelling and simulation environment currently under development, tools are being created to model and simulate various service subsystems and scientific instruments. The interconnections between the power subsystem, comprising solar panels and batteries, and several instruments are illustrated. An interface has been developed between these two types of models (the power system and the instruments), demonstrating the overall power consumption of the scientific instruments.

МУЛТИДИСЦИПЛИНАРНА ПРОГРАМНА СРЕДА ЗА МОДЕЛИРАНЕ И СИМУЛАЦИИ НА СПЪТНИКОВИ МИСИИ: ТЕКУЩО СЪСТОЯНИЕ И ПЕРСПЕКТИВИ

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, много-спътникови мисии, системи от хетерогенни много-спътникови мисии, функционални модели на научни инструменти*

Резюме: *Космическият сектор се развива бързо от единични, масивни спътници към многоспътникови системи. Тази архитектурна промяна се проявява в две основни форми: спътникови съзвездия, които представляват хомогенни (или хетерогенни) групировки, състоящи се от спътници с различни инструменти, маси, размери, енергийни изисквания, както и информационни и комуникационни нужди под управлението на един оператор; и Федеративни спътникови системи (FSS), които динамично споделят ресурси между множество оператори за постигане на съвместни цели.*

Проектирането и управлението на тези силно комплексни орбитални системи прави съвременното компютърно моделиране и симулации абсолютно необходими. Традиционните парадигми за проектиране вече не са достатъчни, за да се справят с мащаба, тясната свързаност между подсистемите и оперативните ограничения в реално време. Поглеждайки напред, следващото предизвикателство в космическото инженерство разчита в голяма степен на разработването на единни, интердисциплинарни симулационни среди. Чрез свързването на различни области – като орбитална механика, термодинамика, телекомуникации и енергийни системи – тези среди позволяват

цялостен анализ на космическите мисии. Същевременно, огромният мащаб на разпределените космически системи генерира експоненциални изчислителни предизвикателства на различните етапи от подготовката на мисията. Разрешаването на тези тесни места изисква високопроизводителни, паралелни алгоритми.

В рамките на развиващата се в момента програмна среда за моделиране и симулации на спътникови мисии се създават инструменти за моделиране и симулация на различни служебни подсистеми и научни инструменти. Илюстрирани са взаимовръзките между енергийната подсистема, състояща се от слънчеви панели и батерии, и няколко инструмента. Разработен е интерфейс между тези два типа модели (енергийната система и инструментите), който нагледно демонстрира общата консумация на енергия от научните инструменти.

Introduction

Recently, various organisations have been planning or implementing the deployment of multi-satellite missions [1, 2]. **Distributed Space Systems** (**DSS**) are developed for diverse purposes, including communication, Earth observation, and geophysical studies (such as Earth and planetary sciences, astronomy, on-orbit servicing, and space situational awareness). The development of such missions covers a wide range of aspects. These span from the required geometry (orbital parameters and, in some cases, the deployment and transition into operational modes) necessary to implement the core concepts, to the optimization of satellite subsystems, individual instruments, the ground segment (control centres and communication), and management software, including operation planning. Selva et al. present a detailed survey of the past, present, and future of **DSS** [3].

Computer software is utilized at various stages, from the preparation to the execution of space missions, with different aspects being simulated to meet ever-increasing requirements. It is important to note that alongside well-known commercially distributed software (such as STK, FreeFlyer, and ASTOS) or open-source alternatives (like GMAT), certain organisations develop proprietary software to address specific needs. Some of these in-house programs extend beyond satellite dynamics simulations [4–7]. While Wertz and Larson suggest using flight-proven software developed and supported by specialized companies, they also highlight that developing proprietary programs allows for a deeper understanding of the underlying challenges [8]. Furthermore, Chagas emphasizes the importance of applying a comprehensive set of modelling and simulation toolkits to develop complex space missions [9]. Ultimately, employing multi-physics environments is crucial to addressing the complex challenges inherent in space mission design [10].

At the Space Research and Technology Institute of the Bulgarian Academy of Sciences (SRTI-BAS), algorithms and software are being developed across various fields related to modelling and simulation. These efforts encompass predicting the motion of multi-satellite systems [11], situational and event analysis [12], functional modelling, and simulating the operations of satellite subsystems and scientific instruments. Recently, research and development have focused on the concept and creation of an interdisciplinary software environment designed for modelling and simulating the behaviour and functionality of multi-satellite missions.

Problem formulation

A concept is being developed for an interdisciplinary software environment dedicated to the modelling and simulation of multi-satellite missions and mission federations operating within near-Earth space. To achieve this, tools are being designed to model various core objects, including space missions, situational events, satellite subsystems, and scientific instruments. Computational instruments and solvers are also being developed to simulate the functionality of these objects.

The software environment under development enables the creation of structurally complex "**System of Systems**" (**SoS**) models. The architecture of these models comprises several distinct layers:

- **Multi-satellite missions:** The highest structural layer, utilized to build models of individual missions and mission federations. It also incorporates models of space debris as an integral part of the space environment in which the satellite missions operate.
- **Space environment:** Computes various model parameters along the satellites' orbits, encompassing both geometric characteristics and environmental properties (such as the atmosphere, ionosphere, and magnetosphere).
- **Situational and event analysis:** Determines the specific time windows suitable for executing satellite operations by solving diverse situational tasks. These tasks are geometric and physical in nature. They are resolved within the scope of an individual mission or federation, or they address proximity evaluations regarding "close encounters" with a member of a given population of space debris.

- **Satellite subsystem modelling:** Focuses on subsystem models, which encompass the Electrical Power Subsystem (EPS), Attitude Determination and Control Subsystem (ADCS), Thermal Control Subsystem (TCS), On-Board Computer and Data Handling Subsystem (OBC/CDHS), Communication Subsystem (Direct-to-Earth (DTE) and Inter-Satellite Links (ISL)).
- **Scientific instrument modelling:** Investigates and creates models for different types of scientific instruments, categorized according to the specific space missions in which they are deployed. Functional models are developed for various instruments based on their operational modes.
- **Dynamic scene modelling:** Develops fundamental models of dynamic scenes used to simulate satellite dynamics within space missions (rendered on a two-dimensional map of the Earth's surface or a three-dimensional representation of the Earth). At this stage, the future development of graphical representations for simulation results related to scientific instrument operations is under consideration.
- **Multiscale modelling:** Develops methods to simulate key features of space missions or individual elements across multiple scales of time and/or spatial resolution.

Depending on the layer to which a given model belongs, specific interfaces are developed to connect it with models from other layers. These interfaces are utilized both during the model composition phase and by the simulation model interpreter during runtime. Furthermore, various software tools and editors are continuously being developed to construct objects based on these models, which themselves undergo ongoing refinement. To simulate the behaviour of the compiled interdisciplinary models, corresponding computational solvers are also being actively designed and advanced.

Software Editor for Compiling Scientific Instrument Models

The configuration, definition, and integration of **Scientific Instrument** (**ScI**) models are managed via the **Scientific Instruments Models Composition Editor** (**ScIMCE**). The **ScIMCE** leverages a library of pre-developed and programmatically implemented payload models. Operationally, the editor employs a dedicated **dialogue box** equipped with diverse controls to select and specify instrument-related parameters. A contextual **list control** presents available instruments tailored to the specific nature of the space mission. Once a particular **ScI** is selected for modelling, its associated parameters are populated via **static text** fields, enabling users to enter data through dedicated **edit controls**. To safeguard against execution errors and block invalid inputs, real-time data validation is enforced based on predetermined parameter domains. Within multi-satellite architectures, the compiled suite of instrument models can be allocated either to a specific satellite or across the entire constellation. This flexibility is vital for heterogeneous missions, where individual space assets are tasked with distinct operational roles.

To construct the scientific instrument models, a structured variable based on a derived data type is utilized, as delineated in Figure 1: The scientific instrument models allocated to a single satellite are stored within an individual column of a two-dimensional array. Consequently, the instrument models for an entire mission are mapped onto an 2D array. The number of columns, M, corresponds directly to the total satellite count. The number of elements within each column, N, is defined by the maximum number of instruments carried by any single satellite within a heterogeneous mission (where different satellites are equipped with diverse payloads and functionalities). Each column incorporates a dedicated zeroth element, which registers the specific instrument count for that individual satellite.

Upon concluding the definition and composition of the **ScI** configuration for a specific space mission, several critical operations are executed:

- The number of threads required to instantiate the **Parallel Scientific Instruments Functionality Simulation Solver (PScIFSS)** is entered into **edit controls**, and the solver itself is instantiated before exiting the dialog. A dedicated subroutine generates these threads, alongside two event objects per thread, which are utilized to synchronise the threads with the parent process. The handles for both the threads and the synchronisation events are stored within a two-dimensional array representing the solver's configuration parameters. Additionally, a global synchronisation event is created to manage inter-thread coordination when fetching the next ScI simulation task or group of such tasks for processing. The number of situational tasks fetched concurrently is determined by the "granularity" parameter, the value of which is configured alongside the thread count.

```
type     satellite_payload ! Tuk se sadarja komplekt ot instrumenti za edin spatnik
  sequence
 UNION
  MAP                    ! Zero element - for control
    character  sat_title*20          ! Satellite name
    integer    sat_id                ! Satellite identification code
    integer    num_instruments       ! Number of instruments disposed on the satellite
  END MAP
  MAP                    ! For next element- description of instruments
    character  instrument_type*32
    character  instrument_name*16
    integer    instr_code            ! Identification code of instrument type
    integer    num_attributes
    real         attributes(1:10)
    logical    situation             ! .true.- instrument depend from some situations-
    integer      id_sit_prob(0:10)   ! serial number of sit.task in all sit.tasks
  END MAP
 END UNION
end type  satellite_payload
```

Fig. 1. Derived data type structure for modelling satellite payloads

- Thread allocation and instantiation: The desired thread count required to initialize the **PScIFSS** is inputted via dedicated **edit controls**, allowing the solver instance to be generated prior to closing the **dialogue box**. A specialized subroutine is invoked to spin up these threads, creating two distinct event objects per thread to facilitate synchronization with the main parent process. The handles for both the execution threads and their corresponding synchronization events are catalogued within a two-dimensional configuration array. Inter-thread orchestration is further managed by a global synchronization event, which coordinates the safe fetching of instrument operational payloads or data blocks during runtime. The volume of data fetched simultaneously by the threads is regulated by a "granularity" parameter, configured concurrently with the execution thread count.
- **Pool of thread creation** Thread allocation and instantiation: The desired thread count required to initialize the **Parallel Scientific Instruments Functionality Simulation Solver** (**PScIFSS**) is inputted via dedicated **edit controls**, allowing the solver instance to be generated prior to closing the **dialogue box**. A specialized subroutine creates these threads and also creates two event objects per thread to facilitate synchronization with the main parent process. The handles for both the execution threads and their corresponding synchronization events are catalogued within a two-dimensional array. Inter-thread orchestration is further managed by a global synchronization event, which coordinates the safe fetching of instrument operational payloads or data blocks during runtime. The volume of data fetched simultaneously by the threads is regulated by a "granularity" parameter, configured concurrently with the execution thread count.
- Prior to exiting the **PScIMCE**, all solver control parameters (such as the thread count and solver configurations) along with the **ScI** data parameters are passed via a global data area (common area) to a wrapper subroutine, which serves as the entry point for the solver's threads. By utilizing different wrapper subroutines - the names of which are passed to the thread-creation subroutine and are essential for instantiating the threads themselves – various configurations of the **PScIMCE** can be implemented.
- All solver parameters and memory addresses containing the **ScI** model parameters, along with the satellite coordinate addresses (and other data), are recorded into a specific derived type structure. This structure is subsequently passed to a subroutine „**add_object**“ possessing polymorphic properties (Fig. 2). This universal subroutine registers solver descriptors into specific groups or "layers", reflecting the comprehensive architecture of space mission simulation models as a **SoS**. The resulting descriptor contains all the necessary situational task data concerning a specific space mission. It is utilized by the simulation model interpreter for a given space mission.

```
common    /c_Payload/num_payloads, ES_payload_adr

ScI_Payload%kind_Payload    = 3              ! 3 - for (Sc)ientific(I)nstruments
ScI_Payload%name            = IVP_par(kod_IVP)%name//"\Instruments"      ! take the name of IVP
ScI_Payload%IVP_index       = kod_IVP    ! serial number of IVP - for satellites coordinates
ScI_Payload% AS_index       = num_AIs    ! Solver's parameters – for interpreter
ScI_Payload%adr_instruments= instrument_descriptor_adr

 CALL add_object(num_payloads, Payload_descriptor_adr        = ScI_payload_adr, &
                                Payload_descriptor_adr_new= ScI_payload_adr, &
                                                level__45= ScI_Payload);
```

Fig. 2. Preparation of the layer descriptor for functional simulations of satellite subsystems and scientific instruments

The total count of all objects within the layer designated as "**level__45**", which encompasses all service subsystems and scientific instrument models, is stored in the **num_payloads** variable. The memory address of this descriptor is recorded in the second position. The variable **kind_Payload** indicates the object type, which is set to 3 for scientific instruments; specifically, this parameter holds a value of 1 for the electrical power subsystem. Furthermore, the **AS_index** variable specifies the sequential index of the thread pool required to start the **PScIFSS** solver. This data is systematically utilized by the simulation model interpreter during runtime.

Parallel Scientific Instruments Functionality Simulation Solver

The **PScIFSS** under development is based on a well-established model previously applied to other computational tools within different layers of the architecture discussed above. Implementing this model is highly appropriate due to the following reasons:

- Computational parallelisation: This is necessitated by the design focus of the software environment, which is tailored for modelling and simulating the functionality of high-dimensional, heterogeneous objects. The large satellite count within a single mission, coupled with the complex models compiled for utility subsystems and scientific instruments, demands significant computational resources. The partitioning of the global computational workload into discrete subtasks is achieved through "**parallelism across the system**", where each thread from the pool executes calculations for an individual satellite. Heterogeneous mathematical models inherently result in irregular computations; these are processed efficiently on multi-core processors by implementing dynamic scheduling based on a "**pool of threads**" programming paradigm [13].
- The entry point of the main **PScIFSS** subroutine is encapsulated within the **wrapper subroutine**; it is passed to the thread pool management subroutine, which subsequently executes the core algorithm that processes the **ScI** models. From an algorithmic perspective, this core subroutine operates as a finite-state machine that accepts an ordered set of situational conditions as an input tape for a given situational task. Each situational condition possesses a unique identifier, which dictates the invocation of the corresponding subroutines responsible for simulating specific **ScI** models.
- **Development of universal utilities:** A unified subroutine is utilized to initialize all solver instances, regardless of whether they employ a single thread or multiple threads. Furthermore, a universal subroutine possessing polymorphic properties has been designed to manage diverse solvers [14]. Concurrently, a "**thread pool union**" programming paradigm is being developed, which enables the simultaneous, parallel execution of multiple parallelised solvers [15]. This approach aims to achieve significantly higher efficiency in computational resource utilization.

Application Example

This section evaluates a practical numerical scenario focusing on the following core objectives:

- **Energy Power System Simulation**: Models solar energy conversion using photovoltaic panels with a total effective surface area of $S = 2\,m^2$ and a conversion efficiency coefficient of $\eta_{eff} = 25\%$.
- **Scientific Instrument (ScI) Simulation**: Evaluates the electrical power consumption of multiple instrument models operating in either standby or operational mode. Two of these instruments transition into operational mode based on the solution of specific situational tasks.
- **EPS–ScI Interface Simulation**: Simulates the dynamic electrical load and power drain imposed by multiple concurrently operating instruments on the power subsystem.

For the purpose of this demonstration, the satellite's orbital parameters are defined as follows: semi-major axis $a = 7{,}200{,}000\,m$, eccentricity $e = 0.001$, inclination $i = 0.0°$, longitude of the ascending node $\Omega = 0.0°$, and argument of perigee $\omega = 0.0°$. A comprehensive event analysis is executed by solving five distinct situational tasks. Situational task ST_1 determines the specific time intervals when the satellite is in the sunlit segment of its orbit, enabling the solar panels to generate electrical energy. Two other tasks, ST_2 and ST_3 (each compiled by two situational conditions), identify the time windows when the sunlit satellite passes over designated ground segments Θ_1 and Θ_2. These ground targets feature geographic coordinates and angular radii configured as $(33°, 1°, 9°)$ and $(66°, 1°, 9°)$, respectively. Finally, situational tasks ST_4 and ST_5 (each incorporating three situational conditions) define the time intervals when the satellite passes over Θ_1 or Θ_2 while positioned within the Earth's shadow (eclipse zone) and with the Moon located below the local horizon (according to satellite).

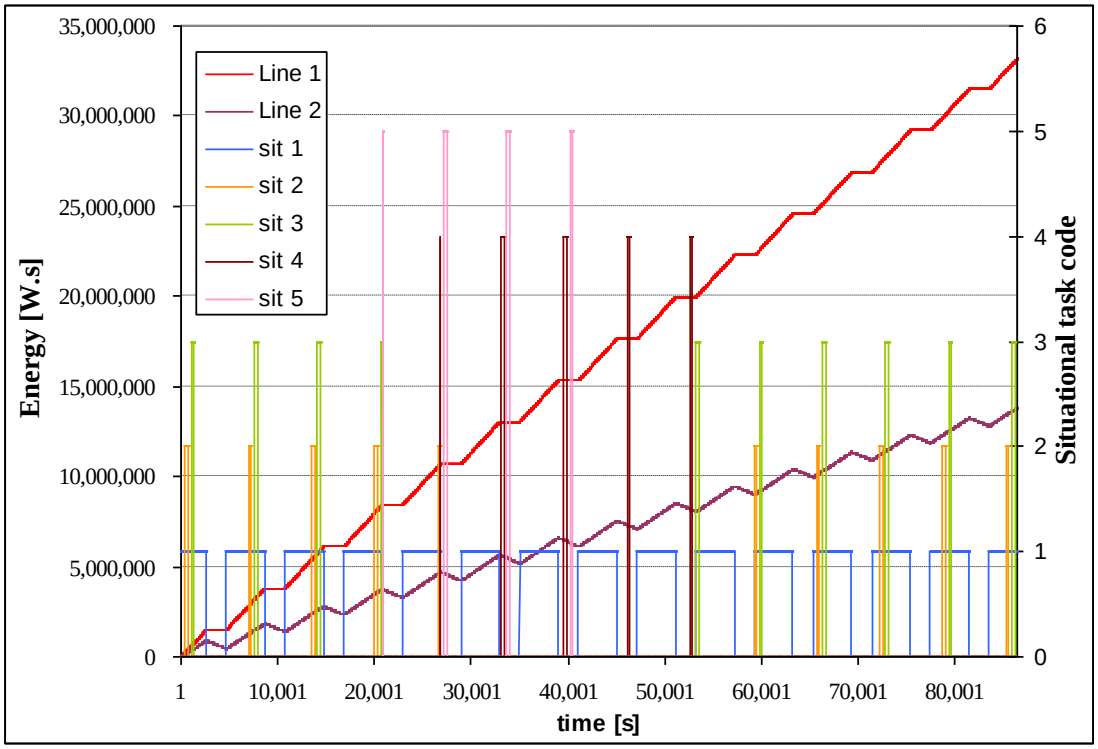


Fig. 3. Line 1 and Line 2 represent the electrical energy generated by the solar panels and its current state as a result of instrument consumption.
Lines sit 1, sit 2, sit 3, sit 4 and sit 5 represent the timing diagrams of the respective situational conditions.

The scenario dynamically models solar energy conversion against the total instrument power demand. In Figure 3, Line 1 presents the converted solar energy received by the panels. Throughout the simulation, a constant (perpendicular) panel orientation relative to the Sun vector is maintained. Efficiency coefficients regarding energy conversion and battery charging are accounted for. Out of six total modelled instruments, four remain permanently in active operational mode with fixed power drains. The fifth asset (an optical instrument operating in the visible spectrum) remains in standby mode and transitions to active operations only when triggered by ST_2 or ST_3. The sixth and final instrument (nominally designated for the infrared spectrum) operates based on the solutions of ST_4 and ST_5. In Figure 3, Line 2 tracks the net residual energy available in the subsystem after incorporating transformation efficiencies and deducting the combined consumption of the active instruments. A secondary vertical axis plots the operational timelines of the five situational tasks, represented schematically using distinct amplitudes corresponding to each task's sequence number.

The primary goal of this demonstration is to validate the communication interfaces established between different models and parallel solvers, alongside utilizing situational task outputs for the deterministic state control of scientific payloads. Further numerical experiments are required to stress-test these frameworks within complex multi-satellite missions, particularly to evaluate the specific characteristics of parallelized models and distributed computational algorithms.

Conclusion and future work

This report presents an initial step in the evolution of the software environment, marking the commencement of modelling specific scientific instruments and developing their interface with the power subsystem. Furthermore, the development of a parallel solver designed to simulate the operations of these instruments has been initiated. To illustrate the capabilities of the developed tools, preliminary results are presented regarding the energy generation from the solar-panel-based satellite power subsystem and the corresponding power consumption by the scientific instruments.

The current models of scientific instruments serve as a foundation for developing an extensive library of models for various spaceborne instruments. Moving forward, it is essential to further develop other satellite subsystems – such as thermal, telecommunication, attitude determination and control, and data storage subsystems – which are currently under investigation. The models for different instrument types will be supplemented by tools for defining instrument command sequences to enable a more detailed simulation of each instrument's functionality and operational states.

References:

1. Esper, J., S. Neeck, J.A. Slavin, W. Wiscombe and F.H. Bauer, Nano/micro satellite constellations for earth and space science. *Acta Astronautica*, *52*(9-12), 2003, pp. 785–791.
2. Millan, R.M., R. von Steiger, M. Ariel, S. Bartalev, M. Borgeaud, S. Campagnola, J.C. Castillo-Rogez, R. Fléron, V. Gass, A. Gregorio and D.M. Klumpar. Small satellites for space science: A COSPAR scientific roadmap. *Advances in space research*, *64*(8), 2019, pp. 1466–1517.
3. Selva D., A. Golkar, O. Korobova, I.L.I. Cruz, P. Collopy, &, O.L. de Weck. Distributed earth satellite systems: What is needed to move forward?. *Journal of Aerospace Information Systems*, *14*(8), 2017, pp. 412–438.
4. Lee, B.S., Y. Hwang, H.Y. Kim and J. Kim, An Integrated Mission Analysis and Planning System for the KOMPSAT-2 Mission Operations. In *SpaceOps 2006 Conference*, 2006, p. 5728.
5. Ambrosio, A.,P. Cardoso and V. Orlando, “Brazilian satellite simulators: previous solutions trade-off and new perspectives for the CBERS program”. In *SpaceOps 2006 Conference*, 2006, p. 5916.
6. Radice, G., R. Wuilbercq and G. Cafiero, “T-smad: a concurrent design tool for space mission analysis and design”. *Frontiers in Aerospace Engineering*, *3*(1), 2014, pp. 1–6.
7. Neto, V.V.G., C.E. Paes, A.J. Rohling, W. Manzano and E.Y. Nakagawa, “Modeling & simulation of software architectures of systems-of-systems”: An industrial report on the brazilian space system. In *2019 Spring Simulation Conference (SpringSim)*, 2019, pp. 1–12.
8. Wertz, J.R., W.J. Larson, Space Mission Analysis and Design. Microcosm Press, Kluwer Academic Publishers, third ed., 1999
9. Chagas, R.A., F.L. de Sousa, A.C Louro and W.G. dos Santos, Modeling and design of a multidisciplinary simulator of the concept of operations for space mission pre-phase A studies. *Concurrent Engineering*, *27*(1), 2019, pp. 28–39.
10. Rabelo, L., L. Low, W. Jung, M. Elkamel, and G. Lee, Enhancing Effectiveness through the Fusion of Machine Learning, VC Simulation, and Agent-Based Modeling. In *IISE Annual Conference. Proceedings* (pp. 1-6). Institute of Industrial and Systems Engineers (IISE), 2024.
11. Atanassov, A.M., Parallel, adaptive, multi-object trajectory integrator for space simulation applications. *Advances in Space Research*, *54*(8), 2014, pp. 1581–1589.
12. 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.
13. Rauber, T., G. Rünger, Parallel Programming: For Multicore and Cluster Systems. Springer, 2010.
14. Atanassov, A., UNIFICATION OF “POOL OF THREADS” CONTROL IN THE FRAMES OF DIFFERENT PARALLEL SOLVERS. Proceedings of SES 2016, 2017, pp. 42–46.
15. Atanassov, A., Method of Thread Management in a Multi-Pool of Threads Environments. *Proceedings of SES*, *2014*, 2015, pp. 241–246.