Proceedings of COBEM 2007
Copyright © 2007 by ABCM 19th International Congress of Mechanical Engineering
November 5-9, 2007, Brasília, DF

A PARALLEL/DISTRIBUTED SIMULATOR APPLIED TO THE MULTI-MISSION PLATFORM SATELLITE ATTITUDE CONTROL

Francisco Carlos de Amorim Terceiro
National Institute for Space Research (INPE), São José dos Campos, SP, Brazil
amorim@dem.inpe.br

Marcelo Lopes de Oliveira e Souza
National Institute for Space Research (INPE), São José dos Campos, SP, Brazil

Abstract. *This work presents a parallel/distributed simulator applied to the Multi-Mission Platform (MMP) satellite attitude control. The MMP is a satellite being developed at the National Institute for Space Research (INPE) following a modern paradigm of satellite development, where the satellite basic architecture can be reused in other missions. The parallel system used to simulate the MMP Attitude and Orbit Control System (AOCS) is composed of two processors and a Real Time Operational System (RTOS). The MMP model includes: the satellite sensors, controller, actuators and dynamics. Tests were realized to control the attitude of the simulated satellite in the Nominal Operation Mode. The test specifications are given by MMP attitude requirements. The preliminary tests have shown that such requirements were met. This is an excellent example of the engineering required to promote the interaction between universities and enterprises in the future.*

Keywords*: parallel/distributed simulation, real time simulation, attitude and orbit control system, multi-mission platform.*

1. INTRODUCTION

This work presents a parallel/distributed simulator applied to the Multi-Mission Platform (MMP) satellite attitude control. The MMP is a satellite being developed at the National Institute for Space Research (INPE) of Brazil, following a modern paradigm in satellite development, where a basic architecture will be constructed to be reused in many other missions (for further information see INPE, 2001). The satellite attitude is the orientation required to the space vehicle physical structure, being determined by the mission, (Fortescue, 2003). The Attitude and Orbit Control System (AOCS) is responsible for acquiring and maintaining this orientation.

Many approaches can be used to help the development of a complex engineering system, in this case an AOCS. The traditional approaches offer a well tested base in which lies the reliability of such techniques, but they are costly and unproductive when compared with some new approaches. Some techniques well tested in others technological areas are now being used to develop space applications, see (Cechticky and Pasetti, 2003) and (Cechticky *et al*, 2003). The XXI century System Engineering allied with updated computer technologies raises the speed of development, lowers the cost, bringing productivity. The recent growth of the Requirements Engineering and Testing Engineering give the needed support to those new approaches to make them trustable and reliable.

In this wide scenario the Modeling and Simulation approach has an essential importance: among many benefits offered to AOCS development, there is the possibility to build scalable and flexible models and simulations that allow the gradual improvement of their levels of realism. For a discussion on previous levels of realism that the simulation of such a space vehicle shall be submitted, see (Milani, 1993).

The choice of an adequate simulator to develop and test the AOCS is essential to mission success, since it allows preliminary analyses during the project phases, and it allows the development team to correct errors as early as possible (Amorim III and Souza, 2006). Some important characteristics to consider when choosing a simulator are: the degree of fidelity to the real system, scalability to support gradual development, flexibility to easily handle changes, and agility to implement changes quickly. To help the quest for assuring the simulator characteristics listed above, this work proposes a simulator that allow the rapid instantiation of the AOCS simulations with the characteristics listed earlier. This idea follows other recent initiatives in the technological areas including the space community, some described in (de Vries, 2003), (de Vries, 2006), (Cechticky *et al*, 2003), (Storch and Liu, 1996) and (Cechticky *et al*, 2002).

This task can be reached using tools off-the-shelf such as the modeling, simulation, code generation and documentation environment MATRIXx/Xmath/SystemBuild/AutoCode/DocumentIt™ of National Instruments Corporation™ and the real time parallel and distributed computing environment iHawk/RedHawk Linux/NightStar™ of Concurrent Computer Corporation®.

The main goal of this work is to propose a simulator that increase the level of realism of the simulations of the MMP AOCS model, where a real time and parallel simulation is made with two processors in the loop, instead of the whole simulation in just one processor. The first step is to move the Attitude Controller to be simulated in a processor separated from the rest of the simulation. Since our main interest is in the Attitude Controller, the separation is needed to improve the development of this module. The importance of the presented simulator is to support the development team with adequate simulations.

2. MODELING AND SIMULATION

What is a model, a simulation and a simulator? Those basic concepts are needed to ingress on the Modeling and Simulation approach. This work will follow the definitions given by (DMSO, 1998) (later extended by Souza and Trivelato, 2003):

a) "—A model is a physical, mathematical, or otherwise logical representation of a system, entity, phenomenon, or process."; b) "—A simulation is a method for implementing a model over time."; and c) "—A simulation is a device, computer program, or system that performs simulations.".

According to (Malyshev *et al*, 1996): "—The complexity of aerospace vehicle control systems and the high cost of even simple flight tests cause the application of simulations in aerospace vehicle control design to be extensive and absolutely necessary.".

They also say that: "—(Naylor *et al*, 1966) suggest that simulation analysis might be appropriate by the following reasons:

1. Simulation makes it possible to study and experiment with complex internal interactions of a given system.
2. Through simulation it is possible to study the effects of environmental changes on the operation of the system by making alterations in the model of the system and observing the effects of these alterations on the system behavior.
3. Detailed observation of the system being simulated may lead to a better understanding of the system and to suggestions for improving it, suggestions that otherwise would not be apparent.
4. The experience of designing a computer simulation model may be more valuable than the actual simulation itself. The knowledge obtained in designing a simulation study frequently suggests changes in the system being simulated. The effects of these changes can then be tested via simulation before implementing then on the actual system.
5. Simulation of complex systems can yield valuable insight into which variables are more important than others in the system and how these variables interact.
6. Simulation can be used to experiment with new situations about which it has little or no information so as to prepare for what may happen.
7. Simulation can server as a "preservice test" to try out new policies and decision rules for operating a system, before running the risk of experimenting on the real system.
8. Simulations are sometimes valuable in that they afford a convenient way of breaking down a complicated system into subsystems, each of which may then be modeled by an analyst or team is expert in that area.
9. Simulation makes it possible to study dynamic systems in either real time, compressed time or expanded time.
10. When new components are introduced into a system, simulation can be used to help foresee bottlenecks and other problems that may arise in the operation of the system.".

Such approach can also be applied to already running systems that are nonstop or are too costly to stop; but that need maintenance or some optimization. For them, its possible to built models of the system and to perform simulations to test changes and find the best way to deal with the problem; and find the minimum cost and time to perform the change. An example of this simulation use can be found at (Amorim III *et al*, 2006).

Beyond all those benefits offered by the modeling and simulation approach, there is the possibility to build scalable and flexible models and simulations that allow the gradual improvement of their levels of realism (Amorim III and Souza, 2007). The modeling and simulation approach is already largely used in the aerospace industry and is also widely applied in all other industry sectors still growing and expanding, specially in the automotive and process industries, as discussed by (Van Doren, 1998).

3. THE MODEL

The first step to ingress in the modeling and simulation is to build models. The modeler should pursue simplicity and accuracy to its models, which are however conflicting characteristics (Amorim III *et al*, 2006). Without contradictions with the definition of the Section 2, which defines a model as a representation, is also possible to define a model as (Meadows *et al*, 1972) does: "—A model is simply an ordered set of assumptions about a complex system, it is an attempt to understand some aspect of the infinitely varied world by selecting from perception and past experience a set of general observations applicable to the problem at hand". Once that is clear there will always be a gap between the model and the object modeled, it is wise to know when to stop modeling and move on to the development to the rest of the simulation solution.

The satellite model used in this work was initially developed for the SACI satellite, as described in (Prudencio, 1997). Later on, it was reused and altered to become adequate to the MMP satellite, as described in (Moreira, 2006). This fact shows the scalability and flexibility, desired characteristics of a modeling and simulation environment. The modeling and simulation environment used to do so was the MATRIXx/Xmath/SystemBuild™. The Xmath™ is the system analysis environment of the MATRIXx™ product family. The SystemBuild™ is a graphical programming environment that uses a block diagramming paradigm with hierarchical structuring for modeling and simulation of linear and nonlinear dynamic systems, (National Instruments, 2004a).

The AOCS model, at the highest hierarchical, includes the components of the satellite attitude and orbit control system, as seen in Fig. 1. The model is divided into blocks that map physical components of a real AOCS, such: Sensors, Attitude Controller, Actuators and Satellite Dynamics.

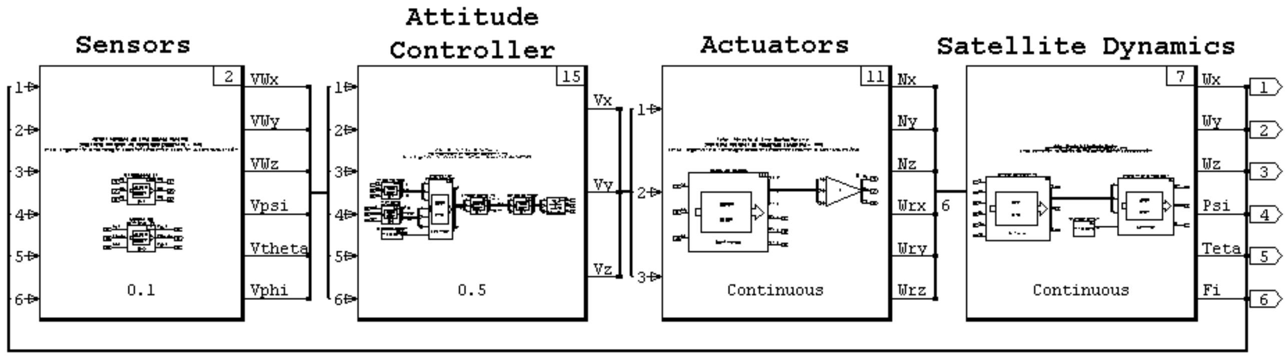


Figure 1: Model used to generate the real time parallel simulator.

3.1 Satellite reference system

The coordinate reference system fixed in the satellite was used to determine the orientation in which the attitude is measured (Moreira, 2006). The coordinate reference systems coincides the axes X, Y and Z with the satellite three principal moments of inertia and the angles Psi, Theta and Phi (attitude angles) are counted counterclockwise around these axes.

3.2 Sensors

The Sensors block receives information from the Satellite Dynamics block, which include the attitude angles and its respective angular velocities. This block includes the model of two sensors: 1) a star sensor, whose outputs are the attitude angles; and 2) a gyroscope, for measuring the attitude angular velocities. After processing the signals it sends the information to the Attitude Controller block.

3.3 Attitude Controller

The Attitude Controller block receives information from the Sensors block, which includes the attitude angles and its respective angular velocities after processed by the Sensors. According to (Moreira, 2006) the attitude controller, with the control laws, will be embedded in a digital computer, so it interacts with the world by electrical voltage values, in its inputs and outputs. The controller was modeled as a time discrete system, since the amplitude quantization and the time delay phenomena were not modeled. The discretization was made using the Tustin rule, and the sampling rate chosen was 10 Hz. The control law used is a Proportional, Integral and Derivative (PID), calculated as described bellow, to the attitude error, measured as the actual attitude angles minus the reference desired attitude angles. After that, the Attitude Controller sends information of the electrical voltage values that should be applied to the Actuators block.

3.4 Actuators

The Actuators block receives the information of the electrical voltage values, from the Attitude Controller block, which will be applied to its set of three actuators. According to (Moreira, 2006) the actuators modeled are three reaction wheels each one placed and aligned with one principal axis of inertia of the satellite. This block has as output the torque produced by each reaction wheel and its respective angular velocities. Those information are then sent to the Satellite Dynamics block.

3.5 Satellite Dynamics

The Satellite Dynamics block receives the information of torque applied to the satellite body by the reaction wheels and its angular velocities, from the Actuators block. According to (Moreira, 2006) this model block includes the dynamic and cinematic equations of the behavior of a satellite orbiting the Earth. The model was split in two blocks: one that represents the satellite dynamics with Euler equations, and the other which includes cinematic equations with Euler angles. After that, this block produces again the attitude angles and its respective angular velocities which are the inputs for the Sensors block and the cycle restarts.

3.6 Synthesis of the Control Law

The model of the control system used in this work is a traditional feedback control loop, as seen in Figure 3, which its Transfer Function (TF) can be read in Eq. (1). The method described here was used to obtain 3 Single Input Single Output (SISO) control system, one feedback control loop for each one the satellite axis, following (Moreira, 2006).

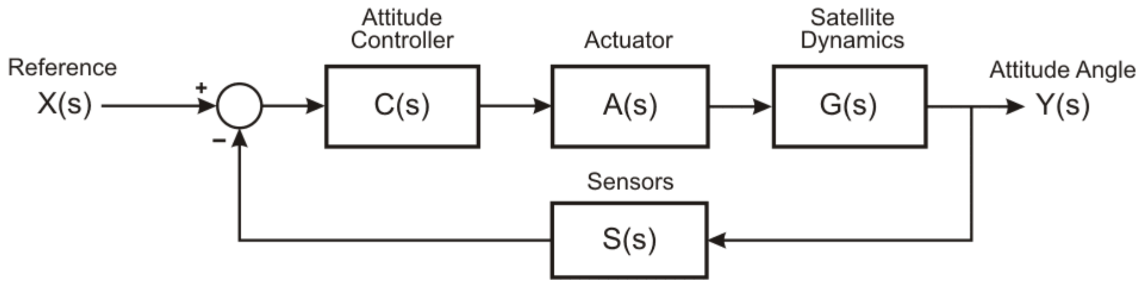


Figure 2: Block diagram of the feedback control system used to synthetize the control law.

$$\frac{Y(s)}{X(s)} = \frac{C(s)A(s)G(s)}{1 + C(s)A(s)G(s)S(s)} \tag{1}$$

The *C(s)* TF of the Attitude Controller was modeled as a PID controller, see Eq. (2). The A(s) TF of the Actuator is a model of one reaction wheel, see Eq. (3). The *G(s)* TF of the Satellite Dynamics was modeled as seen in Eq. (4). The *S(s)* TF of the Sensors was stated as an unitary gain, see Eq. (5).

$$C(s) = \frac{K_P s + K_I + K_D s^2}{s} \tag{2}$$

$$A(s) = \frac{K_V s}{s + \frac{1}{T_V}} \tag{3}$$

$$G(s) = \frac{1}{Is^2} \tag{4}$$

$$S(s) = 1 \tag{5}$$

From the Eq. (2) the parameters K_P, K_I and K_D are, respectively, the Proportional gain, Integral gain and Derivative gain of the PID controller. The parameter K_V is the gain of the reaction wheel and T_V its time constant. The parameter I is the moment of inertia of the satellite related to one axis.

For this model we have a third order characteristic equation, as seen in Eq. (6).

$$s^3 + \left(\frac{1}{T_V} + \frac{K_D K_V}{I} \right) s^2 + \frac{K_P K_V}{I} s + \frac{K_I K_V}{I} = 0 \tag{6}$$

From the statement that a transient response for a unit step input in a third order characteristic equation can be described in terms of:

$$\left(s^2 + as + b\right)\left(s + c\right) = s^3 + (a + c)s^2 + (ac + b)s + bc = 0 \tag{7}$$

where $a = 2\varsigma\omega_n$, $b = \omega_n^2$ and $c = constant$. The parameter ζ is the damping ratio and ω_n the undamped natural frequency of the system response, (Ogata, 1982). From the Eq. (6) and Eq. (7) we have the following equations to calculate the controller gains:

$$K_P = \frac{I}{K_V}(ac+b) = \frac{I}{K_V}\left(2\varsigma\omega_n + \omega_n^2\right) \tag{8}$$

$$K_D = \frac{I}{K_V}\left(a + c - \frac{1}{T_V}\right) = \frac{I}{K_V}\left(2\varsigma\omega_n + \frac{K_I K_V}{I\omega_n^2} - \frac{1}{T_V}\right) \tag{9}$$

From the model of the reaction wheels we have: $K_V = 0.06$ and $T_V = 20$ seconds. For each one the 3 satellite axis the following parameters were chosen: $K_I = 1$, $\varsigma = 0.7$ and $\omega_n = 4 / (T_r\varsigma)$, where the Rise time ($T_r$) is equal to 100 seconds. Using the chosen values in the Eq. (8) and Eq. (9), we have the following controller gains for the axes X, Y and Z:

$$\begin{aligned} K_P &= [40.593;\ 51.785;\ 44.345] \\ K_D &= [454.11;\ 556.93;\ 488.66] \end{aligned} \tag{10}$$

The controllers gains were calculated using a linear model of satellite in which its axes are decoupled between themselves, but the obtained controllers are used to control a satellite model that is nonlinear with coupled axes.

4. THE SIMULATOR

To guarantee agility of the simulator development, the automatic code generator MATRIXx/AutoCode™ was used. It can generate code in C ANSI or ADA 95 for a simulator with real time characteristics, from a partial or a complete model. The process to obtain a simulator using the code generation approach is described in (National Instruments, 2004b). At the end of this process there is a real time simulator of the desired model. Pursuing flexibility and controllability of the code generation process the MATRIXx/AutoCode™ has a Template Programming Language, (National Instruments, 2004c). This language is used to create Template files which hold the information on how the code will be generated, so it is possible to use the same model to generate simulators to different targets and/or different operating systems.

The standard simulator generated by the process mentioned above needs an input file with a time vector; and when the model has some inputs, the simulator also needs a vector with initial conditions at every time instant to all the models inputs. Those data are processed by the simulator to generate an output file with the same format as the input. The output file contains the simulation time instants and the values of all simulation outputs.

4.1 Real Time Systems

Following (Stankovic, 1988), "– In real-time computing, the correctness of a system depends not only on the logical results of the computation but also on time at which the results are produced". Real time computing is not equivalent to fast computing. The goal of real time computing is to attend the time requirements of each task in the computing system. The most important characteristics of a real time system should be the predictability: its functionalities and its time behavior should be as deterministic as necessary to attend the system time requirements.

The motivation for building a real time simulation in this work is to prepare the MMP AOCS simulation to communicate with real devices, which leads to a simulation called hybrid simulation or Hardware-In-the-Loop (HIL) simulation. In this level of realism simulated models are substituted for real devices. Once the simulation is already running in real time the substitution process is conducted smoothly.

4.1.1 Scheduling

The scheduler is an object that is responsible to allocate processing time resources to the many tasks that compete for this resource. In a real time system the scheduler should be deterministic and predictable. The MATRIXx/AutoCode™ builds a generic scheduler as part of the generated real time simulator. This scheduler performs overall direction and control of inserting inputs, scheduling tasks, posting outputs, and dispatching the tasks that perform the work of the real-time system. It operates on the principle of rate-monotonic scheduling (National Instruments, 2004a). The scheduler generated by the MATRIXx/AutoCode™ can be tailored to some specific target or operating system by means of the template file. A Template file was developed to make the generated source code operate adequately with the real time parallel and distributed computing environment iHawk/RedHawk Linux/ NightStar™.

The iHawk™ is a developing platform for real time parallel and distributed systems. It includes a workstation computer with parallel processors, and Real Time Clocks. The RedHawk Linux™ is a Real Time Operating System (RTOS) that implements many features for real time system development, including the shield feature. This makes it possible to shield a processor to any other activity but those which are assigned to be executed in that processor. The NightStar™ is a set of tools to develop and test real time parallel and distributed systems. It includes its own scheduling tool, NightSim™ that is a real time tool that provides a graphical user interface to the frequency based scheduler and performance monitor services (Concurrent Computer Corporation, 2002).

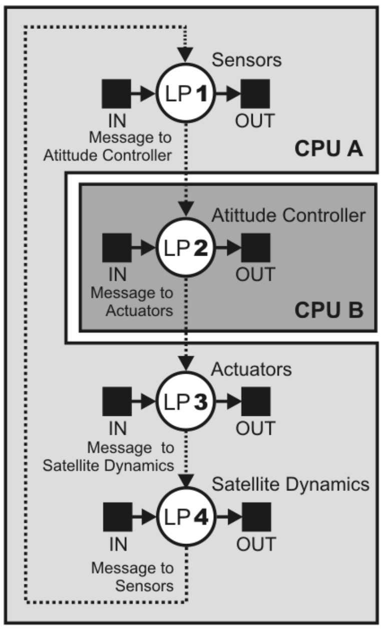
4.2 Parallel and Distributed Simulations

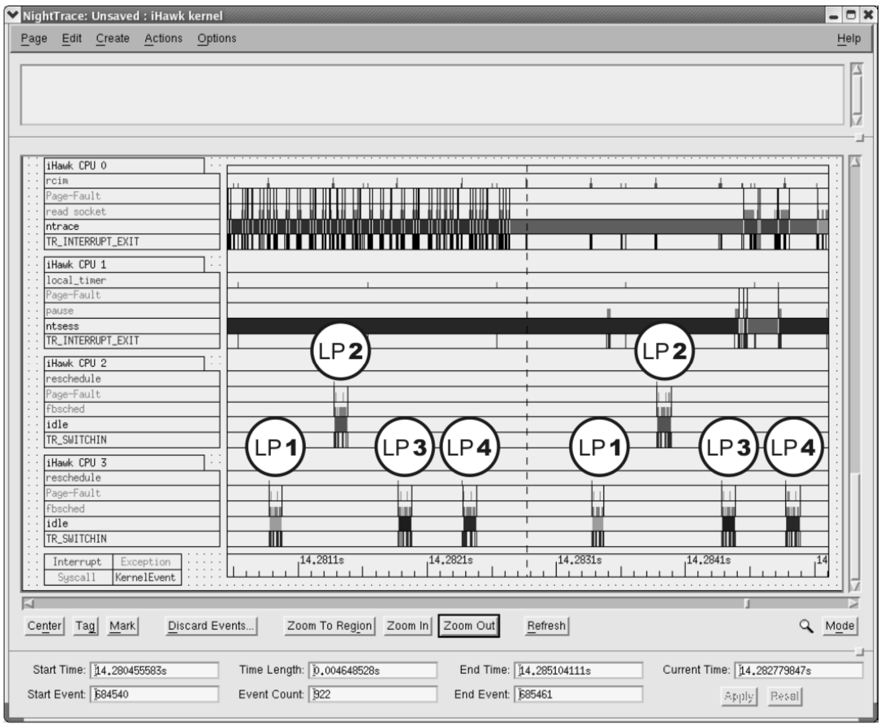
The reason which leads this work to face with parallel and distributed simulations was the need to isolate the attitude controller from the rest of the simulation preparing it to be loaded in the processor target chosen for the MMP mission. Before moving to this step, as we described earlier, is a good practice test and evaluate the attitude controller. The Attitude Controller was simulated in one processor and the other three models were simulated in a distinct processor. This kind of simulation deals with ways to use multiple processors in the same simulation (Perumalla, 2006). The whole system composed of simulators and others entities that together perform simulations are then called a Parallel/Distributed Simulation System (PDSS).

The spatial parallel system is the scheme used in this work. In this structure, the model is partitioned along spatial dimensions: at every time instant all model partitions should be simulating at the same time instant. Only this requirement introduces two other preoccupations to the PDSS: the causality, and the synchronization. For further information see (Perumalla, 2006) and (Fujimoto, 2000). Each simulator into the PDSS is called a Logical Process (LP), and each LP contains its own individual state variables. To obtain a spatial parallel system from the MATRIXx/AutoCode™ it was necessary to improve the developed Template file, for real time simulation, to generate code that support communication between the LPs, obtaining LPs with exactly the same structure of a complete simulator in one process, i.e., every LP has an input file and an output file. To get the PDSS working correctly, the LPs should exchange information that is quantitatively correct at the adequate time. To achieve this requirement the PDSS should provide means of communication and ways of causation and synchronization between LPs. The communication method implemented here was the message passing technique.

4.3 The AOCS Simulator

The source code of the simulator was generated by MATRIXx/AutoCode™ from the models developed in the MATRIXx/SystemBuild ™ environment. The programming language chosen was the C ANSI. The simulator is composed of four LPs, one for each model block: the Sensors, the Attitude Controller, the Actuators and the Satellite Dynamics. The source codes were compiled using GNU Compiler Collection (GCC). The LPs were loaded in the iHawk™ platform. The NightStar/NightSim™ was used to scheduling the LPs and to make them be executed at the required time. Two processors were used to perform the simulation and both were shielded to tasks that do not belong to the PDSS. One processor was in charge to execute the Attitude Controller LP and the other to execute the LPs: Sensors, Actuators and the Attitude Dynamics. Beyond the NightStar/NightSim™ scheduler, each LP has its own scheduler generated by MATRIXx/SystemBuild/AutoCode™ to be responsible for its internal tasks, since every LP is autonomous and independent from the others. A diagram of the PDSS structure can be seen in Figure 3 (a). and the respective execution of two simulation steps in Figure 3 (b). The Figure 3 (b) shows a captured screen of the NightStar/NightTrace™ tool and shows all process being executed in the 4 processors of the iHawk™ platform in one of the performed tests. Notice that the processor 3 is executing only the LPs: Sensors, Actuators and Satellite Dynamics. The processor 2 executes only the Attitude Controller LP. The processors 0 and 1 is in charge of all other kinds of tasks such as the operating system, any other software running at the time, and the response to the input and output devices.





Start Time: 14.280455583s
Time Length: 0.004648528s
End Time: 14.285104111s
Current Time: 14.282779847s
Start Event: 684540
Event Count: 922
End Event: 685461

(a) (b)

Figure 3: (a) A diagram of the PDSS structure; (b) Process executed in the iHawk™ during two simulation steps.

The simulator execution run as a cyclic executive: at any determined time instant only one LP is being executed; and after all the LPs of the simulator have been executed the cycle restarts. The simulation runs until all LPs have simulated trough all time instants from its time vector, contained in every LP input file.

5. RESULTS

An specified attitude maneuver that takes one attitude angle (the Psi angle) initially from 30º to 0º ± 0.05º, and 0º/s ± 0.001º/s, in $t_f - t_0 < 180$s was used to test the simulator. Figure 4 presents a comparison between results from: 1) the virtual time simulation, made in MATRIXx/SystemBuild™; and 2) the real time parallel simulation, made with the four LPs generated with MATRIXx/AutoCode™, executed in a real time parallel and distributed computing environment iHawk/RedHawk Linux/NightStar™. The three top graphics in Figure 4 show the attitude angles (Psi, Theta and Phi) respectively, and the three bottom graphics show the angular velocities (Wx, Wy and Wz) respectively, around the satellite principal moments of inertia X, Y and Z. Figure 5 shows the errors between the compared simulations, with graphics organized in the same structure as Figure 4. The errors seen in Figure 5 are produced by the different approaches used for each simulation: the virtual time simulation is adopted as the standard; and the real time parallel simulation is compared with this standard.

The satellite axis in which the maneuver was performed (the Psi angle) had the largest transient error among the 3 axes, less than -1.7º at the beginning of the maneuver; and it stabilizes close to zero degrees after 100s. The largest error was expected in this axis, since it is the axis in what happens most of the action, and the angles assume larger values. The other two axes, with Theta and Phi angles, had minor errors. The Theta angle, see Figure 4, in the real time simulated stabilizes with offset errors, not bigger than 0.012º. In Wy angular velocity, yet in Figure 4, both simulation shows a bias rate close do -0.062º/s, around the Y axis due to the orbital motion. After the stabilization period of 150s the biggest difference between the two simulations is found in the error graphics Theta, see Figure 5, which is smaller than 0.013º. This difference is much smaller than the 0.05 degree pointing specification (INPE, 2001), and the other results of the MMP AOCS simulation meet the specifications of repositioning the MMP satellite of up to 30º in less than 180s.

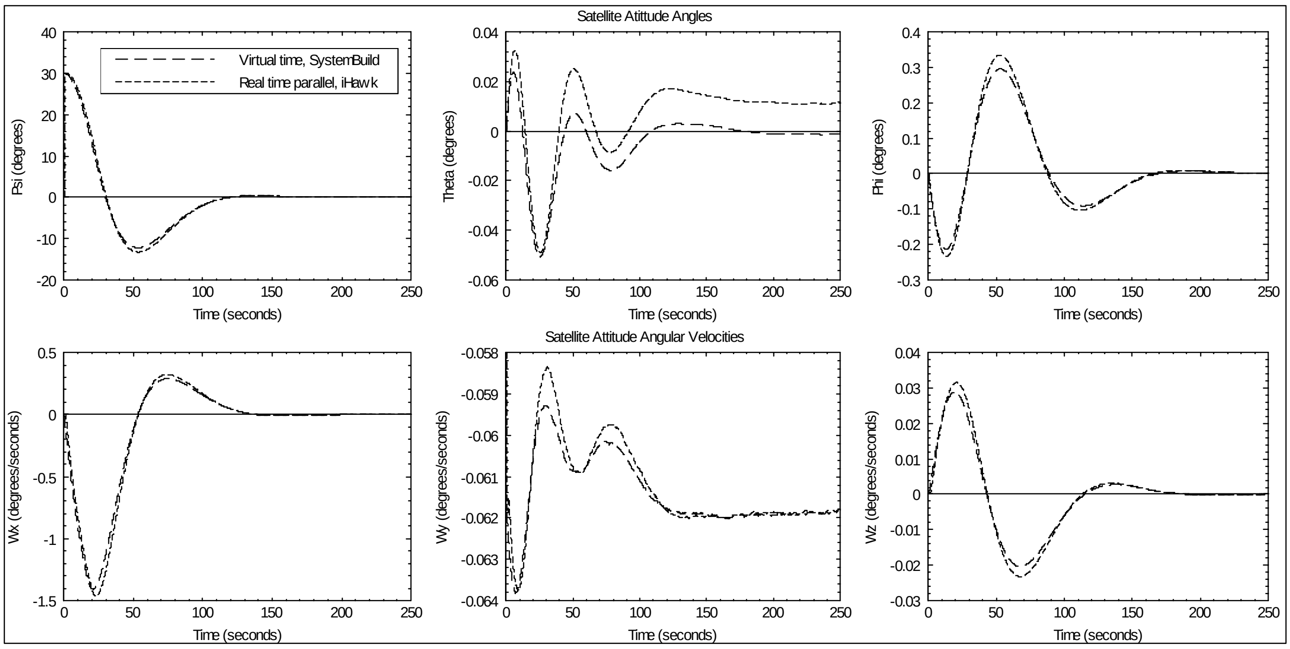


Figure 4: Comparison between real time and virtual time simulations.

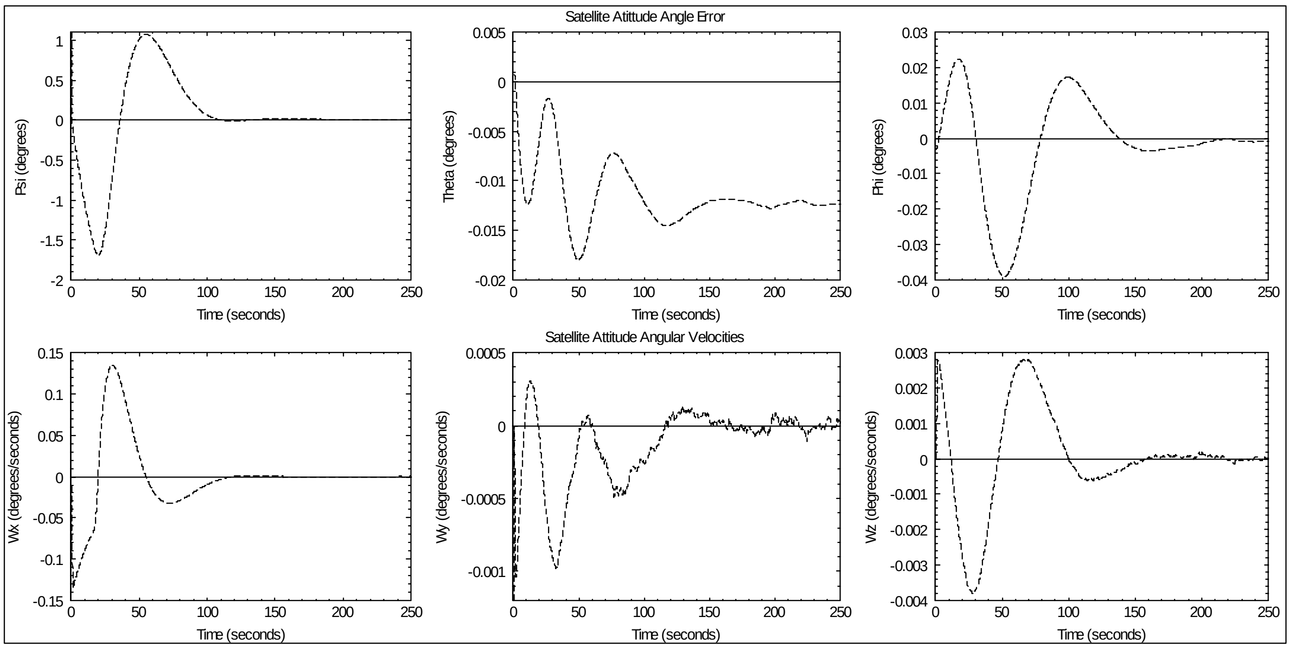


Figure 5: Errors between real time and virtual time simulations.

6. CONCLUSIONS

This work presented a parallel/distributed simulator applied to the AOCS of the Multi-Mission Platform (MMP) satellite, and adequate for testing it in real time. The parallel system used to simulate it is composed of two processors and a Real Time Operational System (RTOS). The MMP model includes: the satellite sensors, controller, actuators and dynamics. Tests were realized to control the attitude of the simulated satellite in the Nominal Operation Mode. The test specifications are given by MMP attitude requirements. The preliminary tests have shown that such requirements were met. The differences between the performed virtual and real time simulations are acceptable and were expected. The theory and methods exemplified in this work with an aerospace application can perfectly fit the needs of other technical communities from industries of all sectors. The modeling and simulation approach can be used when any of the needs described in Section 2 are desired. This approach is specially useful to reduce cost and time in system development, being a great opportunity to promote interaction between universities and enterprises.

7. ACKNOWLEDGMENTS

The first author thanks the National Council for Scientific and Technological Development - CNPq of Brazil for supporting him during this research. The second author thank the National Institute for Space Research - INPE/DMC of Brazil for supporting him and the Laboratory of Computational Environments for Simulation, Identification and Modeling of AOCS (LABSIM2) where this work is being done.

8. REFERENCES

Amorim III, F.C., Kuga, H.K. and Lopes, R.V.F., 2006, “Modeling the Internal Temperature of an Autoclave Using Parameter Estimation”. In XII Latin-American Congress on Automatic Control (CLCA 2006), Salvador, BA, Brazil.

Amorim III, F.C. and Souza, M.L.O., 2006, “Simulação Paralela e em Tempo Real do Sistema de Controle de Atitude da Plataforma Multi-Missão em Modo Nominal”. In XIII Colóquio Brasileiro de Dinâmica Orbital (CBDO), Bertioga, SP, Brazil.

Amorim III, F.C. and Souza, M.L.O., 2007, “A Framework for Real Time and Parallel Simulation of the Attitude and Orbit Control System of the Multi-Mission Platform Satellite”. In 6^o Brazilian Conference on Dynamics, Control and Their Applications (Dincon’07), São José do Rio Preto, SP, Brazil. (submitted).

Cechticky, V., Montalto, G., Pasetti, A. and Salerno. N., 2003, “The AOCS Framework”. In K. Fletcher and R. A. Harris, editors, ESA SP-516: 5th ESA International Conference on Spacecraft Guidance, Navigation and Control Systems.

Cechticky, V. and Pasetti, A., 2003, “Generative Programming for Space Applications”. In ESA SP-532: DASIA 2003.

Cechticky, V., Pasetti, A. and Schaufelberger, W., 2002, “A New Approach to Software Development for Embedded Control Systems”. In MSy, pages 75–83,Winterthur, Switzerland.

Concurrent Computer Corporation, 2002, “NightSim User’s Guide”. Concurrent Computer Corporation, Pompano Beach, FL, US.

de Vries, R., 2003, “EuroSim Simulation Framework”. In ESA SP-532: DASIA 2003.

de Vries, R., 2006, “EuroSim - New Developments on a Proven Real-Time Simulator Environment”. 9th International Workshop on Simulation for European Space Programmes (SESP 2006).

DMSO, 1998, “DoD Modeling and Simulation (M&S) Glossary”. Technical Report DoD 5000.59-M, Defense Modeling and Simulation Office (DMSO).

Fortescue, P., Stark, J. and Swinerd, G., 2003, “Spacecraft Systems Engineering”. John Wiley & Sons, Ltd, 3rd edition.

Fujimoto, R. M, 2000, “Parallel and Distributed Simulation Systems”. John Wiley & Sons.

INPE, 2001. “Multi-Mission Platform Attitude Control and Data Handling (ACDH) Subsystem Specification”. Instituto Nacional de Pesquisas Espaciais, A822700-SPC-001/05, São José dos Campos, SP, Brazil

Jefferson, D.R., 1985, “Virtual Time”. ACM Transactions on Programming Languages and Systems, 7(3):404–425.

Malyshev, V.V., Krasilshikov, M.N., Bobronnikov, V.T., Dishel, V.D., Leite Filho, W.C. and Ribeiro, T.S., 1996, “Aerospace Vehicle Control: Modern Theory and Applications”, Instituto de Aeronáutica e Espaço (IAE), FAPESP, São José dos Campos, SP, Brasil.

Meadows, D.H., Meadows, D.L., Randers, J. and Behrens III, W.W., 1972, “The Limits to Growth”. Universe Books. New York, NY, USA.

Milani., P.G., 1993, “A New Architecture for the Simulation and Testing of Satellite Attitude and Orbit Control Systems, Hardware and Software Description”. World Congress of the International Federation of Automatic Control (IFAC), Sidney, Australia, 5:161–164.

Moreira, M.L.B., 2006, “Análise, Projeto e Simulação de um Controle Discreto para a Plataforma Multi-Missão e sua Migração para um Sistema Operacional de Tempo Real”. Master’s Thesis, Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, SP, Brasil.

National Instruments, 2004a, “MATRIXx: Getting Started Guide”. National Instruments Corporation, Austin, Texas, USA.

National Instruments, 2004b, “MATRIXx: AutoCode User Guide”. National Instruments Corporation, Austin, Texas, USA.

National Instruments, 2004c, “MATRIXx: Template Programming Language User Guide”. National Instruments Corporation, Austin, Texas, USA.

Naylor, T.H., Balintfy, J.L., Burdick, D.S. and Chu, K., 1966, “Computer Simulation Techniques”, New York, NY, USA, John Wiley.

Ogata, K., 1982, “Engenharia de Controle Moderno”. 1. ed. Prentice-Hall do Brasil, Ltda., Rio de Janeiro, Brasil.

Prudencio, S.V., 1997, “Simulação Digital em Tempo Real de um Sistema de Controle de Atitude Magnético Autônomo de um Satélite”. Master’s Thesis, Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, SP, Brasil. (Approved in 1997 and published in 2000).

Perumalla, K.S., 2006, “Parallel and Distributed Simulation: Traditional Techniques and Recent Advances”. Proceeding of the 2006 Winter Simulation Conference (WSC’06), 2006.

Souza, M.L.O. and Trivelato, G.C., 2003, “Simulators and Simulations: Their Characteristics and Applications to the Simulation and Control of Aerospace Vehicles”. In XII SAE Brazil Congress, São Paulo, SP, Brazil.

Stankovic, J.A., 1988, “Misconceptions About Real-Time Computing: A Serious Problem for Next-Generation Systems”. Computer, 21(10):10–19.

Storch, M.F. and Liu, J.W.S., 1996, “DRTSS: a Simulation Framework for Complex Real-Time Systems”. Page 160, Los Alamitos, CA, USA. IEEE Computer Society.

VanDoren, V.J., 1998, “Simulation Simplifies ‘What-if’ Analysis”. In Control Engineering, 45(9): 68-76.

9. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.