A NEW ARCHITECTURE FOR THE SIMULATION AND TESTING OF SATELLITE ATTITUDE AND ORBIT CONTROL SYSTEMS. HARDWARE AND SOFTWARE DESCRIPTION

P.G. Milani

Space Mechanics and Control Division, National Institute for Space Research - INPE, P.O. Box 515, 12201-270, Brazil

Abstract. Brazil has just launched its first artificial satellite. As part of that effort is the development of a laboratory that enables one to perform simulations in real time, with hardware in the loop, with the critical phases of the mission being matched as closely as possible. This article describes the architecture that was planned for this lab, its implementation and the improvements that were made based on the team experience since its conception. The hardware is based on a three axis dynamic simulator, a supermini computer, Sun and Earth simulators and auxiliary equipment. The general architecture of the system is presented and its relationship to the software explained. The software is being developed in two different ways. Both the approaches are operational in real time and they are explained in more detail in the text. An interface between Matlab and the simulation hardware is being developed as well and it helps the tasks of testing and simulating in non-real time.

Keywords. real time computer systems, simulation, attitude control.

1. INTRODUCTION

Brazil has just launched it's first satellite. It will be a series of four: two for environmental data collection and two for remote sensing applications. The last two of them due to the nature of the mission have more restrictive specifications on the attitude control system then the first ones. It is necessary to control rate to 8 x 10-3 deg/s and a pointing accuracy of 1 deg. These last two satellites will be injected in a 98 deg. inclination low altitude tranfer orbit and then transferred to a Sun synchronous orbit at 640 km altitude with zero eccentricity. Attitude control will be provided by a 4 Nms momentum wheel, a set of 3 magnetic torque coils and monopropelant thrusters. A digital Sun sensor, an infrared Earth sensor and a 3 axes magnetometer will be used for attitude determination.

2. DEGREES OF REALISM

There are many levels of realism to which a satellite Attitude and Orbit Control System must be tested for (Rios Neto and Fleury, 1984). The first and possibly the most used (and simple) approach is the non-real time, software only simulation, in which all the system is coded in software and is executed in a computer with no interfacing to external hardware. The second level is software simulation in real time, i.e.,it is executed at least as fast as the real system producing results at the same rate[1]. This imposes restrictions on the control laws that can be used and mainly restrictions on the integrator for the Dynamics of the satellite (fig. 1).

The next level is the real time static simulation with hardware in the loop. At this level of realism only the Control Law block of Fig. 1 is implemented in hardware and "taken out" of the simulation computer. The rest of the system remains as before. This requires the development of special interfaces in order to communicate with the Control Law computer.

Other degrees of realism require the use of dynamic simulators that are used to move (or stimulate) phisically the real sensors that are going to be used in the AOCS, where the output of the Dynamics module commands the dynamic simulators via appropriate hardware interfaces. The simulators then stimulate the Sensors which feedback to the Control Law which in its turn closes the loop. This approach enables the gradual substitution of Sensors Modules by their respective hardware counterparts as soon as they become available either from own development or acquired from the market.

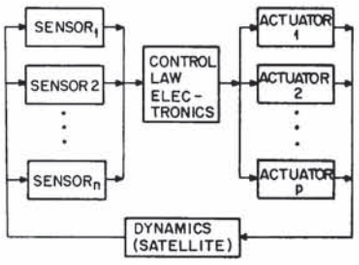

Fig. 1. The control loop of the simulation system

It is not necessary (but it is possible) to substitute the actuators modules by their hardware counterparts but just to monitor their behavior during the simulation through the use of dedicated hardwares that represent them. In the case of a Gas Jet system, for example, each thruster is represented by a led in a pannel. When a maneuver is executed the corresponding leds blink accordingly. It isn't practical (if feasible) to feedback the torques and forces of such an actuator. The same reasoning applies to magnetic actuators.

[1] Due to the discrete nature of digital simulation this means that the computer must be able of computing the complete cycle of Fig. 1 in a discretization period of the simulated system.

The Sensors and Actuators can be stimulated not only physically but also electrically. This will be seen in more detail later on.

3. HARDWARE DESCRIPTION

The structure of the complete hardware can be seen in Fig. 2. It is composed of a VAX 11/780, an Array Processor FPS 5410, a Dynamic Simulator 53M2-30H, a Vaxlab workstation, 5 workstations, an Horizon Simulator with interface electronics and a Sun Simulator with Sensor interface electronics. The Multi-Standards & the FPS/C Interfaces aren't shown. Also available for some specific kind of simulation there are: a one axis and a three axis air bearing tables and a one axis servo table.

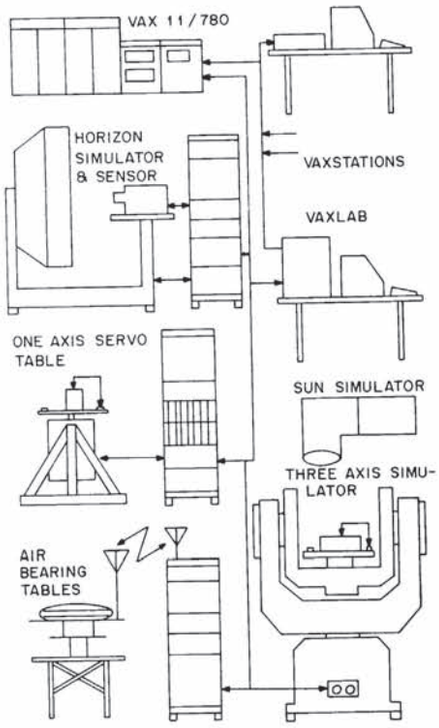

Fig. 2. Description of the Simulation.

In order to cope with the caracteristics of different degrees of realism, the hardware must be connected properly for each situation. In the software kind of simulation, in non real time, it is possible to execute the whole software in any one of the CPUs of the structure or in many of them (distributed system). In this last case each separated module of the complete software executes concurrently in a different CPU and communicates and synchronizes to the other modules of the network via special software resources.

In the static simulation case with hardware in the loop, it becomes necessary to include the Multi-Standards Interface (MSI) and a Control Unit that will execute the software of the Control Law in the control loop, as in Fig. 1 or 2. The MSI has been designed to be able to convert from any kind of sensor interface to the Digital's DR11-W interface. Up to the moment it has been implemented interfaces that comply to the ESA standard, a serial synchronous data interface for satellite use. The MSI may be easily adapted to other serial/parallel interfaces due to the existence of an internal multplexed bus that is shared by all the different sensors/actuators circuits. At this level of simulation the Control Law Unit, usually the AOCS computer, "sees" the AOCS components through the lines that connect it to the simulation computer. At this level of simulation the AOCS components are only software modules. This is enough to test the Control Law Unit with respect to quantization errors of the data words, delays in the transmission of2the words, noise, and the Control Law itself among other things.

It has been considered that a Vaxstation configured with a number of data acquisition interfaces (Vaxlab) was an appropiate choice for the needs of the tasks involved. Ethernet was required for all the workstations of the network in order to cope with the distributed processing requirement. These stations have been chosen because VMS is an operating system that the working group is used to and it provides a set of tools that can be used for comunications and synchronism for real time simulation.

When in dynamic simulation with hardware in the loop another degree of realism can be added to the system through the use of physical stimulation of the sensors by environmental simulators. This approach is somewhat questionable today and it can be considered valid depending on many factors like team experience, resources available and realism of simulation required for any particular attitude or orbit manuver.

There are 3 simulators that can be used at the laboratory of the Division of Space Mechanics and Control (DMC) of INPE. There are a static Sun simulator, a three axis dynamic simulator (Contraves 53M2-30H) and a dynamic Horizon (or Earth) simulator that is under development.

It can be seen from Fig. 2 that the output of the Dynamics module provides control information to the dynamic simulators. This control occurs according to the satellite attitude and orbit that is being propagated (integrated) inside the computer. This way it has been developed in house an interface that is able to control the three axis dynamic simulator. This interface connects the array processor FPS 5410 to the Contraves control pannel. The array processor is a remainder of the original simulation system as well as the Vax 11/780 and they are about to be replaced.

This FPS/C interface is basically a protocol converter between two general purpose interfaces provided by the array processor and the three axis simulator manufaturers. It converts from negative to positive logic, assures that data strobe and data-ready/ready-for-data pulses are timed according to specifications and guarantees perfect line matching at the I/O connectors (Milani,1990). There is a special software to drive this interface (Diehl,1990).

The Horizon simulator is being developed in house.It will be able to dynamically simulate the Earth horizon with a positioning precision of at least 10E-2 deg. Due to the nature of it's control system low transfer rates of command words are to be used and a simple RS-232c interface is going to be used to connect it to the computer. The Horizon sensor is fixed to an appropriate suport at the Horizon simulator.

The Solar simulator is a fixed one and it provides a collimated beam of 20 cm diameter light that is directed to the solar sensors. These last ones will be mounted on special fixtures fixed to the inner axis of the 3 axes dynamic simulator. The solar simulator is positioned according to critical phases of the mission so that the AOCS can be tested for these cases.

The last degree of realism may be obtained when all (or almost all[2]) of the Sensors software modules are replaced by their respective hardware and are

[2]There are some sensors that are not practical to replace. For example: Earth magnetic field sensor.

connected to the Control Unit via their original interfaces and the commanded signals are sent via the MSI to the Actuator software modules. As mentioned before it is not practical to simulate actuators but in some cases it may be necessary to integrate a Reaction Wheel or another actuator in the loop. This can be done by using an Air Bearing Table (ABT) for one or more axis, being the dynamics of this ABT replaced in the software part of the Dynamics module.

Many times it may not be feasible (physically or economically) to utilize physical simulation. This can be overcome by using electrical stimulation of the AOCS components. The disadvantage of this procedure is that the physical part of the component is not tested and as is known from practice this may hide problems. Again, the decision criteria to which kind of stimulation should be used must be based on the team experience and knowledge of the AOCS components.

The previous hardware structure can be implemented in many ways. The use of a modular concurrent software permits each module, if desired, to run on a different computer or workstation. This way the workstations can be anywhere in the facility with a different group working on each one of them. Each group can use then the same idea of different degrees of realism for the development of its own module according to the project needs. In case it is necessary a special hardware to simulate a special condition, like a thermal or vacuum chamber, it may be easily integrated to the simulation environment. A central computer would make this impossible.

4. SOFTWARE DESCRIPTION

There are two basic approaches for the simulation software: a concurrent and a sequential one. A third possibility may be used in order to test individually the AOCS components in a manual way. The sequential type of simulation software is possibly the most used one. Many times it is coded in Fortran without the use of special software tools. A description of an implementation of this approach would be a large program composed of a serial sequence of code that would perform the same functions of the modules of Fig. 1 (Rios Neto et al., 1987, 1987, 1988). The advantages of this approach are ease of development, no need for special software resources, easy interchange of data between the internal modules and no need to manage the simulation, all these being a direct result of the way the program is coded.

There are also many disadvantages: each module of the AOCS is simulated at a time and it becomes more difficult to replace the code by the corresponding hardware due to timing considerations. The program is more difficult to test because it is much larger than single modules. There is no great advantage in the division into parts of this kind of program and distributing it on many computers, actually there will be an extra delay (communication between the computers) that will have to be accounted for.

The distributed type of simulation software can be coded in a variety of languages including Fortran. Since distributed software is more natural to personel with computer science background there are a number of languages other than Fortran that would make this kind of simulation easier to handle. This is not a limitation, though, as long as the operating system can provide some basic communication and synchronization tools. Not all the available commercial operating systems provide you with a comprehensive set of primitives that enables one to program a real time distributed system. The basic needs in order to permit this kind of simulation are:

1) the possibility to create a number of concurrent processes where each of the processes is a program developed in the usual way,
2) the availability of a mechanism of transference of data between3the processes,
3) the availability of a mechanism for the control and synchronism of the processes.

It would be interesting also that the use of these just mentioned software tools were as transparent as possible to the control engineer.

As an implementation example one can look at the software as is described by Nunes (1991). The general idea behind the simulator is to map each of the blocks (as in Fig. 1) to processes in the computer and utilize the communication mechanisms to represent the arrows. In each process a different program is executed even if written in different programming languages. Only the convention of the message exchanged must be respected. In this system "mailboxes" were used for this communication and a Monitor program was used to control them. The VMS, the operating system of the Vax computer family provided these tools which were accessed by the Fortran written modules of the simulator from the Real Time Lybrary (RTL) of the computer. Nunes also says that the modularity of the software enables one to program it and test it individually and that they may be written in Fortran by control specialists which may not be familiar with the sofisticated techniques of real time programming.

There are two other processes that participate of this programming environment: the first one is the Monitor which creates the mailboxes, inialize the execution of the processes and receives alarm messages during simulation. A Post-Processor is the other and is used for exhibiting the results.

In order to create this concurrent software one may use two teams that participate of the development phase at two different points in time. The first team implements a general skeleton for the whole system where each module is dumb. The general architecture would be respected though and any satellite with the characteristics of Fig. 1 could be considered for implementation. This group would also develop the Monitor and the Post-Processor modules, which create and communicate to the special software environment. This should be done by the computational background group. The other team must be composed of control engineers that fills in the skeleton with the code for the sensors, actuators, etc.

As mentioned previously the modules may be distributed in different computers and the messages between them can be sent via the network. It does not matter to the simulation itself if all the modules are resident in only one computer or if they are distributed in many other ones. In case there is one module that requires massive processing or special I/O for any particular reason (e.g. the Dynamics Module) then this module may be separated in a dedicated computer (or workstation).

Also, in the case of replacing the software by hardware, all that is necessary is to call from a dumb module the driver for the corresponding hardware interface. This is equivalent to filling in the module with a description of the AOCS component but putting real hardware in the simulation loop. On the other hand, when making static simulation with hardware in the loop, that is, only the Control Electronics module (and all of its interfaces to the sensors and actuators) is in hardware, then there must be drivers to handle all the interfaces. This driver calls the software modules and connects them to the interface lines. Care must be taken though because some components cannot be simulated in software as fast as the hardware interface requires. (This is particularly true with the ESA standard). This can be compensated for by an over-simulation technique in

which many intermediate results are prepared beforehand and the right one is chosen only at the time that the hardware interface interrupts the corresponding software module. The driver software for this particular case is under development at the moment at INPE's laboratory.

The distributed approach has its disadvantages also. It is more difficult to mannage it's operation. It depends on a close interaction of two groups with different backgrounds and the interfaces between these two groups must be clearly stated and this depends on the hardware for that simulation, also. The main advantages are its' modularity, testability and ease of adaptation to different missions.

Putting it all together

The diagram of Fig.3 shows a general case in which there are two Sensors (Se_2 & Se_3) phisically stimulated by two simulators (Sm_2 & Sm_1) which are commanded by their respective interfaces (FPS/C & HSI) that in their turn are driven by their respective drivers in software. The other sensor (Se_1) is only simulated in software and is only connected to the Control Law Unit (like the other two sensors) via it's Esa Standard serial synchronous line. In this last case though, the stimulation of the sensor is done in software, directly from the Dynamics module. The connection of the Sensor (Se_1) to the Control Law Unit is done via the MSI and it's driver. Figure 3 also shows two paths from the Control Law to the two Actuators (Act_1 & Act_2) also via the MSI. The modules Dynamics, Actuators 1 & 2 and Sensor 1 compose the distributed software environment and may be in any computer of the network. The same may happen to the interfaces (FPS/C, HSI, MSI). The logical structure of the whole is maintained and controlled by the Monitor module.

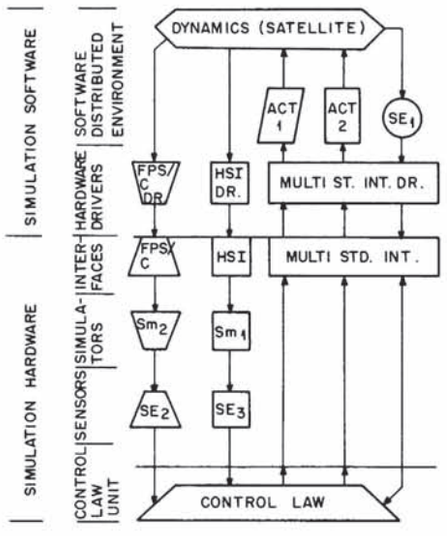

Fig. 3. Software and hardware simulation diagram.

5. MATLAB INTERFACE

Once there is a hardware interface (FPS/C) and the driving software, it has been built an additional software interface that enables the connection of MATLAB to the three axis dynamic simulator. The initial objective was to bring real time data of the simulation to the interactive environment of MATLAB. All the commands that are available through the driver interface (that is: reads and writes, rates and positions, inhibit settings, etc.) can be directly accessed via the keyboard (or an M-file) during a session of MATLAB. This enables the commands of posátion, rate and acceleration as well as reads without the need of writing a program, compiling it, linking it and only then being able to interact with the simulator.

MATSIM proved to be too slow for real time simulations. In the Vax 11/780 in which it is resident up to the moment, it presented a command transfer rate of 2 to 3 Hz only. This is probably due to the interpretive nature of MATLAB. If it were possible to compile M-files into executable code, it would probably be much faster and the advantages of Fortran programs could be brought to this environment (Milani, 1992).

It is also under consideration the development of MATLAB software interfaces to other simulation equipment. Even in non real time simulation this kind of operation proved to work quite well for manual testing with automatic logging of the results and with the availability of a number of graphics resources.

6. CONCLUDING REMARKS

It has been presented a hardware and software architecture for real time dynamic simulation applications. From the beggining of its operation the system is being used with the classical sequential type of software. Nowadays due to the low cost of workstations it has become more interesting to decentralize the simulation system and distribute the tasks between the computers and working groups. Characteristics of the system were described and their advantages/disadvantages examined.

The software was presented following two different lines: a sequential and a distributed approach. For both cases the architecture of the system was detailed to the level of the driving interfaces. The MATLAB/simulator software proved to be slow but very practical for manual use.

7. BIBLIOGRAPHY

Diehl, J.B. (1990). "Utilização das Rotinas para Controle do Simulador Contraves 53M2-30H Através da Interface GPIOP do Processador Matricial FPS 5410". INPE-5153-RPI/234, Internal Report, DMC.

Fleury, A.T; Rios Neto, A. (1984). "Control Systems Dynamic Verifications: First Progress Report". Spar Technical Report RML/009/84/24. February, 9.

Milani, P.G. (1990). "FPS/C. Uma Interface de Comunicação Entre o Processador Matricial FPS 5410 e o Simulador Dinâmico Contraves 53M2-30H". Internal Report. INPE/DMC.

Milani, P.G. (1992). "MATSIM: Um Software de Interface Entre o MATLAB e o Simulador Contraves". Internal Report. INPE/DMC.

Nunes, D. (1990). "Um Simulador Distribuído de Sistemas de Controle de Atitude e Órbita em Tempo Real". 1o. Simpósio Brasileiro de Tecnologia Aeroespacial. Instituto Tecnológico de Aeronáutica - ITA, São José dos Campos, SP, Brasil, 27 to 31 of August.

Rios Neto, A. et al. (1987). "Requisitos de Usuário do 'Software' de Simulação de Atitude em Tempo Real". INPE, Internal Report, DMC, June.

Rios Neto, A. et al. (1987). "Documento de Requisitos de 'Software' de Simulação de Atitude em Tempo Real". INPE. Internal Report, DMC, June.

Rios Neto, A. et al. (1988). "Documento de Projeto Preliminar do Software de Simulação de Atitude em Tempo Real". INPE. Internal Report, DMC, April.