Proceeding of the 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space:
i-SAIRAS 2003, NARA, Japan, May 19-23, 2003

The SPDM Task Verification Facility: On the Dynamic Emulation in One-g Environment using Hardware-in-the-Loop Simulation

M. Doyon, J.-C. Piedbœuf, F. Aghili, Y. Gonthier and E. Martin

Canadian Space Agency, 6767 Route de l'Aeroport, St-Hubert, Quebec, Canada, J3Y 8Y9
michel.doyon@space.gc.ca

Keywords: Hardware-in-the-loop Simulation; Space Robotics; Dynamic Emulation

Abstract

To verify all robotic tasks involving a space robot interacting with environment, such as the Special Purpose Dexterous Manipulator (SPDM), one should appeal to a simulation technique because the space robot cannot operate in an 1-g environment. However, to simulate dynamical behavior of a robot interacting with environment creates difficulties due to complexity of the physical phenomenon involved during the interaction. In this work we present an hardware-in-loop simulation (HLS) technique, where a simulation of the space robot dynamics is combined with emulation of the contact dynamics by using a rigid robot performing contact task. The rigid robot is not dynamically or kinematically equivalent to the space robot, but it is controlled so that its endpoint dynamics replicates that of the space robot. Simulation and experimental results given from implementation on a six degrees of freedom manipulator are presented.

1 Introduction

Canada's contribution to the International Space Station (ISS) is the Mobile Servicing System (MSS) [1]. A major component of the MSS is the Special Purpose Dextrous Manipulator (SPDM). While the Canadarm 2 (Space Station Remote Manipulator System) will assemble the ISS, the SPDM will be required for performing maintenance tasks. Essentially, the SPDM will manipulate the Orbital Replacement Units (ORU), the components of the ISS systems replaceable on orbit. The SPDM will operate directly connected to the ISS or to the tip of the Canadarm 2. Both the Canadarm 2 and the SPDM are tele-operated by an operator located inside the ISS. Due to the important flexibility in the Canadarm 2/SPDM system, all insertion/extraction tasks involving the SPDM will be done using only one arm with the other arm grasping a stabilization point.

The cost and risks associated with the execution of robotic tasks around the ISS require that all procedures be verified on earth prior to their execution in space. The Canadian Space Agency (CSA) is completing the development and verification of the SPDM Task Verification Facility (STVF). One of the main technical challenges with the STVF is the verification of the feasibility of the insertion/extraction tasks. Simulation is a viable tool to validate functionality of a space manipulator [2]. A faithful model of the space robot is available, but accurate modeling of contact dynamics is difficult to obtain due to the complex nature of the physical phenomenon during the interaction.

The main difficulty in emulating a space robot on ground is the fact that space manipulators cannot support their own weight on earth. Different possibilities exist for the ground emulation of a space robot. The first one is to use a flat floor as done for Shuttle Remote Manipulator System or European Robotic Arm validation. However, the limitation to a plane is not representative of real contact task. A second possibility is to use a scaled-down version of the space manipulator. While attractive in theory, this is very difficult to realize in practice especially for a robot having flexibility. The third option is to use counterweights to build a system that will be dynamically equivalent to the space robot as

done by MD-Robotics for the engineering test of the SPDM. This is an interesting option but matching the frequency response of the space robot is difficult. In addition, the SPDM is mounted itself on the Canadarm 2 which is very flexible. A self-balancing system is not able to represent the flexible motion of the base. The last option is to use hardware-in-the-loop simulation (HLS) as done by the CSA [3],[4]) but also by DLR [5] and NASA [6].

The HLS simulator consists of a rigid robit with its control, a simulator for Canadarm2/SPDM dynamics and a visualisation engine. The SPDM operator sends joystick commands to the SPDM simulator which predicts a corresponding motion response. The resulting SPDM endpoint motion then becomes a setpoint for the rigid robot controller.

The contact forces are measured using force/moment sensors and fed back into the simulator to allow the dynamic simulation engine to react to external contact forces. This concept is very flexible since it can accommodate vibration of the space robot base or other phenomena. It can also be used to represent different space robots. The main difficulty in HLS is to have good performance while keeping the system stable in free space and in contact. This type of simulation creates instability problems similar to the case of force control with a force reflecting master/slave system in teleoperation.

The CSA is using a cartesian feedback linearisation technique with acceleration input ([7],[8]). This gives good performance and good stability. However, it requires a stiff robot with high tracking accuracy, a good torque controller and the ability to implement or access the robot controller at the lowest level to achieve a fast sampling time.

2 Controller for Dynamics Emulation

The rigid robot (terrestrial robot) control synthesis is of prime importance in the HLS concept. Since the idea is to replicate the dynamics of the space robot with the terrestrial robot performing the contact task, the control algorithm must be such that the controlled terrestrial robot is transparent in the frequency band of interest for the analysis required. The control approach presented in this paper consists in forcing the terrestrial robot to behave like the simulated space robot by commanding its Cartesian acceleration. Cartesian position/velocity feedback is used in addition as a corrector to improve the system response within the bandwidth of interest. The control approach is presented in details in input [7].

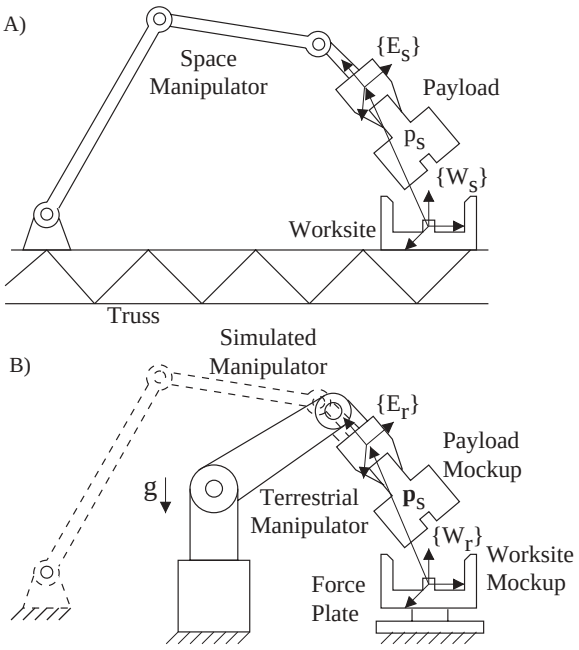

Figure 1: Space Manipulator (A) and Terrestrial Manipulator (B) performing contact task.

Figure 1A illustrates a space manipulator handling a payload. The space robot is typically flexible and redundant. Figure 1B illustrates the control approach. It shows a ground robot performing a contact tasks while emulating the dynamics of the space robot. The contact forces are measured and fedback to the space robot controller (see figure 2.

In reality, the dynamics of robot manipulators is highly nonlinear. In addition to its inherent dynamic coupling, more severe phenomena such as joint friction and actuator dynamics usually limit the ability to properly model and control manipulators. In the case where friction is dealt with using model-based friction compensation or through local feedback of joint torque sensor signals, the method still applies. In that case, neglecting joint friction compensation errors, the generic dynamic equation for a rigid ground robot (terrestrial robot) is

$$\mathbf{M}_G(\mathbf{q}_G)\ddot{\mathbf{q}}_G + \mathbf{h}_G(\mathbf{q}_G, \dot{\mathbf{q}}_G) = \mathbf{B}_G\mathbf{u}_G \qquad (1)$$

where

$$\mathbf{u}_G = \left\{ \begin{array}{c} \boldsymbol{\tau}_G \\ \mathbf{F}_G \end{array} \right\}$$
$$\mathbf{B}_G = \left[\begin{array}{cc} \mathbf{1} & -\mathbf{J}_G^T \end{array} \right]$$

Here, $\mathbf{q}_G$ are the joint coordinates, $\mathbf{M}_G$ is the inertia matrix, $\mathbf{h}_G$ is a vector containing all nonlinear effects, $\mathbf{J}_G$ is the manipulator Jacobian, $\boldsymbol{\tau}_G$ is the vector of control force/torques and $\mathbf{F}_G$ is the vector of force/torque applied to the end effector.

To apply the acceleration command control, the ground robot is first decoupled in Cartesian space through joint-level nonlinear feedback as presented in [9]. The nonlinear feedback law to achieve Cartesian decoupling is given by

$$\boldsymbol{\tau}_G = \hat{\mathbf{h}}_G + \hat{\mathbf{J}}_G^T \hat{\mathbf{F}}_G + \hat{\mathbf{M}}_G \hat{\mathbf{J}}_G^{-1} (\boldsymbol{\gamma} - \hat{\dot{\mathbf{J}}}_G \dot{\mathbf{q}}_G) \qquad (2)$$

where $\boldsymbol{\gamma}$ represents the new control input to the linearised robot. Applying the linearisation law (2) to the robot dynamics (1), the motion is then described by

$$\ddot{\mathbf{X}}_G = \boldsymbol{\gamma} + \boldsymbol{\Delta} = \mathbf{J}_G \ddot{\mathbf{q}}_G + \dot{\mathbf{J}}_G \dot{\mathbf{q}}_G \qquad (3)$$

where $\boldsymbol{\gamma}$ is identified as the desired tip acceleration of the rigid robot and $\boldsymbol{\Delta}$ represents the error in linearisation. The desired acceleration is defined by

$$\boldsymbol{\gamma} = \ddot{\mathbf{X}}_S + \mathbf{K}_v \left(\dot{\mathbf{X}}_S - \dot{\mathbf{X}}_G \right) + \mathbf{K}_p \left(\mathbf{X}_S - \mathbf{X}_G \right) \qquad (4)$$

where $\mathbf{X}$ represents a combination of the angular and linear motion. For the error in orientation, the rotation matrices must be used to compute an orientation error vector.

3 Software Development and Implementation

The control system development and implementation is done using Symofros environment [11],[12]. As shown in figure 3, Symofros is used from the development stage up to the real-time implementation phase of the project. The main goal behind this approach is to minimize as much as possible any hand recoding and to go from the simulation used in the development to the real-time code used in the control of the rigid robot.

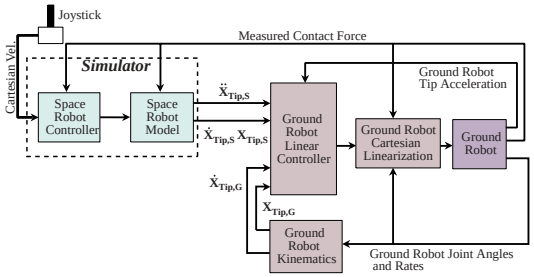

Figure 2: Cartesian Feedback Linearization with Acceleration Controller

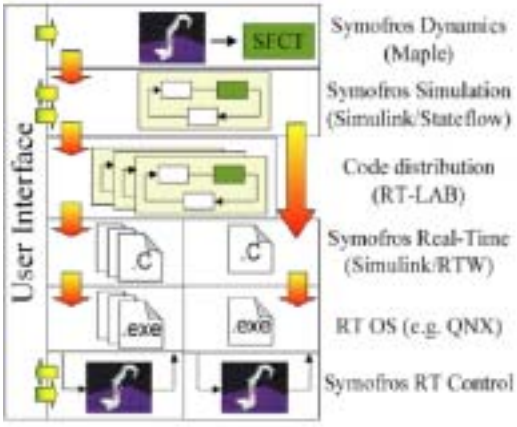

Figure 3: Symofros Architecture

3.1 Concept

The main philosophy behind the code development is the division of the system in functionalities (or modules). Each module can be developed and tested quickly and independently in simulation using Symofros-Simulation under Simulink. The modules are categorized in different levels of libraries that facilitate unit testing, configuration management and integration. To develop the controller, a realistic model of the robot has been developed using Symofros-Dynamics. Therefore, we have a calibrated (kinematically and dynamically) simulation model of the robot having the same input/output signals as the real robot. This allows to run a pure non real time simulation of the overall system to first assess that a given module can be integrated in the main diagram. The next step consists in generating the code for the real time target but for pure simulation with no hardware to test different simulation runs or controller gains with real time performance. Afterward, we replace the pure simulation blocks with the real hardware block and we generate the code to run hardware in the loop in real time.

The main objective for Symofros architecture was to design a real time, distributed and open environment with the possibility to run in real time or in non real time, not the opposite. Targetting a real time environment influences the approach taken, for example the first level hierarchy of the Simulink diagram represents the various CPUs on which the diagram will be splited and the code generated.

With the approach described above, a single SIMULINK diagram represents the simulation, the implementation, the code running on the real time platform and as a byproduct the report describing the diagram can be generated. In terms of configuration management, this approach has a lot to offer. We initially define the input/ouput data exchanges between the CPUs. Then, the functionalities (controllers, safety system, etc) are components that are sourced from libraries representing modules that have been unit tested.

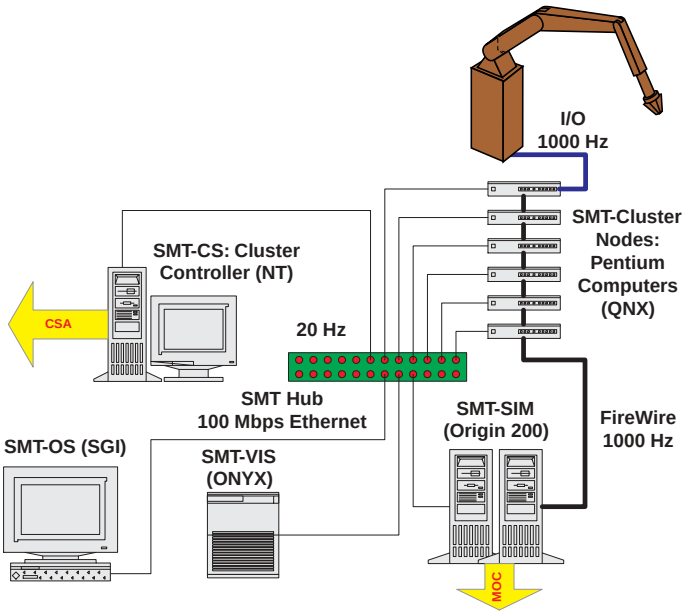

Figure 4: Computer Architecture for the Space Robot Dynamics Emulation

3.2 Implementation

We are using a heterogeneous computer environment to maximize the use of existing CSA simulation facilities (see figure. 4). The controller of the terrestrial robot is running on a cluster of six Pentium computers running under QNX. The real-time simulator of the space robot uses the MSS Operation and Training Simulator (MOTS). The dynamic engine of MOTS is running on a SGI Origin 200 machine with four processors while the visual engine is running on a SGI Onyx machine with four processors. The Origin 200 is running as a slave node to the cluster of Pentium computers. The connection between the Origin 200 and all the Pentium is via a Firewire link. Since the visual and the human interface do not require high speed communication, a fast ethernet link is used. The architecture is very modular since the user can decide on which CPU each module will run. For more details on system architecture hardware and software see [11].

The allocation of the nodes is as follows. The low-level controller is running on the Node 1. This node also contains a timer card for the hardware synchronisation of all CPUs. Node 1 also contains most of the Input/Output (I/O) of the robot, the remaining I/O being on Node 2. Node 1 runs both the low level torque controller and the simulation model. The user selects the mode of operation, simulation or hardware, from the console and it can be changed on the fly during the operation. A safety machine with an emergency position controller is also running on Node 1. Node 2 runs the high level controller (the feedback linearisation) and a state machine to manage the operational modes of the rigid robot. This node has two firewire cards: one for the communication with other computers of the cluster and one for the communication with SPDM robot simulator. Node 3 contains the low level controller of the OTCME (the robot end-effector). The gravity compensation and the forward kinematics, both using the Symofros model of the robot, are also calculated on this node. Node 4 contains all the necessary code for the visual environment, namely control of the camera views. Finally, Node 5 runs a Symofros model of the space robot that can be used instead of the SMT-Sim model. Both space robot models, SMT-Sim and Symofros, can run at the same time. The actual one tracked by the SMT robot is selected from a switch in the console. Node 6 is used for compilation and as a backup node.

The fastest sampling frequency in the system is 1000 Hz. This is a MOTS requirement to ensure stability of the space robot simulator since a Euler integration method is used. Other parts of the system do not required to run at such a fast rate. Our Symofros architecture is able to handle multi-rate systems in a multi CPUs environment (i.e. many different rates on many different CPUs) while maintaining interCPU synchronisation. While the code running on the Pentium

Figure 5: STVF Testbed Manipulator

cluster is directly managed by Simulink, the code running on MOTS uses its owns process management system developed by CAE Electronics. This system also allows multiple time band simulation.

A standard Graphical User Interface provided within Simulink and RT-Lab allows the user to control the execution of the simulator. A key element is the tools available for data acquisition: real time data logging of signals, snapshot of all the signals from all the CPUs and online parameter tuning. In addition, an application program interface (API) allows the interface to third party GUIs, e.g. Labview or custom designed GUIs.

4 Experimental Results

4.1 Square Peg Insertion

Since an off-the-shelf industrial manipulator could not meet our requirements, a custom hydraulic manipulator (see figure 5) was build by International Submarine Engineering Ltd, a Canadian company with significant experience in delivering hydraulic manipulators for submersible and terrestrial applications. The key issues for the ground robot design for HLS and for SPDM emulation can be found in [13].

A **square** peg insertion case was tested experimentally (see figure 6)and is presented below. The tolerance between the square peg and the worksite is 0.25mm. Figure 7 shows the position of the simulated SPDM and of the terrestrial robot (SMT) while figure 8 shows their respective velocities. Finally figure 9 shows the resulting experimental forces measured during the insertion.

Figure 6: Square Peg Insertion

In this scenario, the worksite is mounted on a rigid force plate to accurately measure contact forces and moments. This means that we have a very stiff robot contacting a rigid worksite, controlled in such a way to replicate the dynamics of a flexible system. The peg is positioned a few centimeters above the worksite with a small misalignment. The SPDM is commanded to move in the axial direction only.

Initially, the robot is moving in free space. Then, after time equal 8 seconds, the end effector contacts the worksite for the square peg (in the z direction). The insertion itself happens between time 8 seconds and 30. An extraction occurs after time equals 30 seconds and the peg is fully extracted at time 48 seconds. After about 20 s, the peg hits the side of the worksite and the force moment accommodation scheme of the SPDM realigns the peg to permit the insertion. The various forces exerted during the insertion are friction forces (mainly x and y direction) and as well the applied force in the z direction. The force is stabilised around 200 N (z direction). It can be seen on figures 7 and 8 that the terrestrial robot (SMT) is tracking the space robot (SPDM) with a good accuracy. The Experimental results errors are presented in table 1. The system positionning specifications for the project are 6 mm and 0.5 degrees. Although the various velocities and forces seem low, they are typical of operational velocities of the SPDM in space.

Finally, from figure 9, the vibrations observed are the

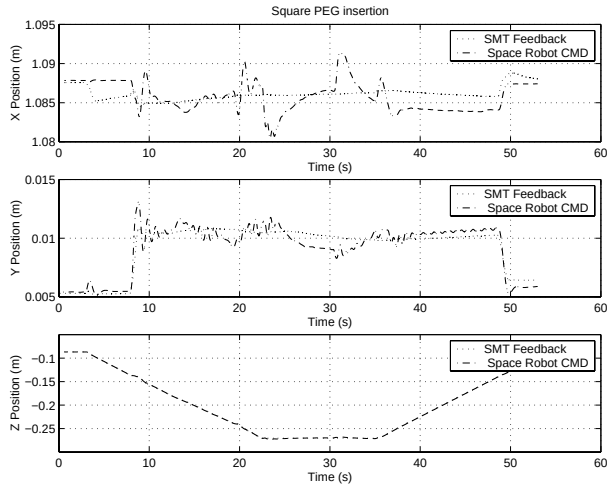

Figure 7: Experimental Positions for Square Peg Insertion

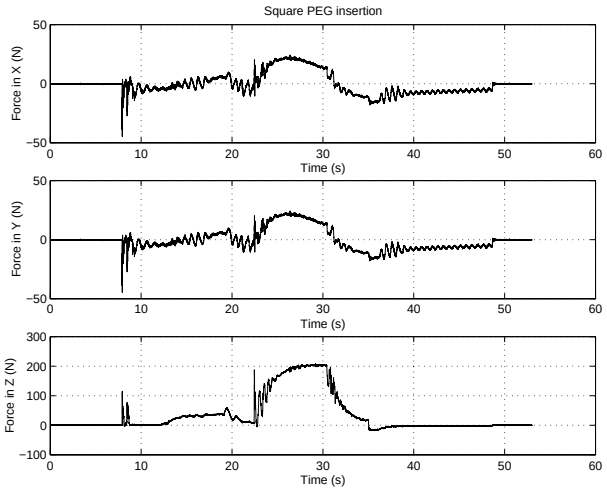

Figure 9: Experimental Forces for Square Peg Insertion

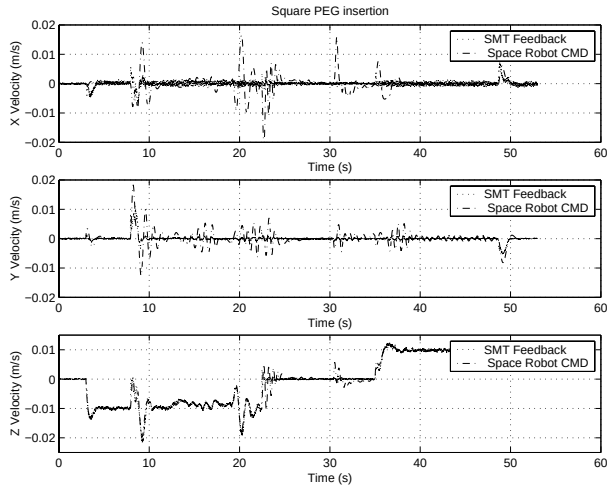

Figure 8: Experimental Velocities for Square Peg Insertion

Direction	Max	Mean	STD
X Position (mm)	5	1.5	1.1
Y Position (mm)	3	0.50	0.4
Z Position (mm)	3	0.57	0.4
X Squew angle (deg)	0.09	0.30	0.02
Y Squew angle (deg)	0.23	0.061	0.05
Z Squew angle (deg)	0.1	0.021	0.02
X Velocity (m/s)	0.02	0.002	0.002
Y Velocity (m/s)	0.012	0.001	0.001
Z Velocity (m/s)	0.012	0.0006	0.0009
X Angular Velocity (deg/sec)	0.011	0.0009	0.001
Y Angular Velocity (deg/sec)	0.019	0.0013	0.002
Z Angular Velocity (deg/sec)	0.012	0.0007	0.001

Table 1: Errors

ones of the space robot itself with the current force moment accomodation gains. This problem will be solved by adjusting the space robot force moment accomodation gains (as the SMT robot is emulating the behavior of the Space robot performing the contact task).

4.2 Bandwidth Limit

There is a limit in the bandwidth to which the Ground robot can emulate the Space Robot. Figure 10 present the results of the emulation of the space robot by the ground robot while injecting band limited white noise (up to 3 Hz) as an input command to the ground robot. This limit is mainly caused by the various hardware components of the ground robot system, namely the friction, the weight of the ground robot (1650 kg), the compressibility of the oil that limit the performance of the torque controller as well as the limitation is due to the dynamic model estimation errors.

5 Conclusions

We demonstrated that a space robot can be emulated by a terrestrial manipulator using a hardware-in-the-loop simulation technique. We have shown that the controller needs to cancel the dynamics of the terrestrial robot. This is done through a cartesian feedback linearisation technique with

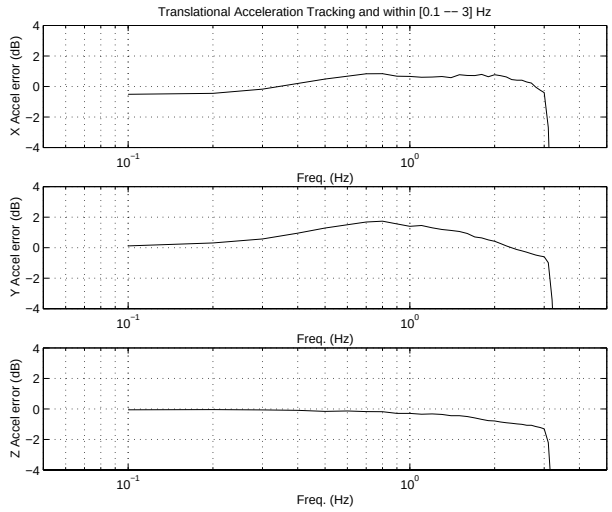

Figure 10: Experimental Free Space Bandwidth

the acceleration of the space robot endpoint as input. We also discussed the critical aspect of the implementation on a real robot. We have used Symofros, allowing quick prototyping and real-time implementation. We demonstrated the validity of our approach by presenting experimental results showing a square peg insertion task.

References

[1] M. Stieber, S. Sachdev, and J. Lymer, "Robotics architecture of the mobile servicing system for the International Space Station," in *Proceeding of the 31st International Symposium on Robotics (ISR 2000)*, (Montreal, Quebec), pp. 416–421, Canadian Federation of Robotics, May 2000.

[2] O. Ma, K. Buhariwala, N. Roger, J. MacLean, and R. Carr, "MDSF- A generic development and simulation facility for flexible, complex robotic systems," *Robotica*, vol. 15, pp. 49–62, 1997.

[3] J.-C. Piedbœuf, J. de Carufel, F. Aghili, and E. Dupuis, "Task verification facility for the Canadian special purpose dextrous manipulator," in *1999 IEEE International Conference on Robotics and Automation*, (Detroit, Michigan), pp. 1077–1083, 10-15 May 1999.

[4] F. Aghili, E. Dupuis, J.-C. Piedbœuf, and J. de Carufel, "Hardware-in-the-loop simulations of robots performing contact tasks," in *Fifth International Symposium on Artificial Intelligence, Robotics and Automation in Space* (M. Perry, ed.), (Noordwijk, The Netherland), pp. 583–588, ESA Publication Division, 1999.

[5] R. Krenn and B. Schäefer, "Limitations of hardware-in-the-loop simulations of space robotics dynamics using industrial robots," in *Fifth International Symposium on Artificial Intelligence, Robotics and Automation in Space* (M. Perry, ed.), (Noordwijk, The Netherland), pp. 681–686, ESA Publication Division, 1999.

[6] S. Ananthakrishnan, R. Teders, and K. Alder, "Role of estimation in real-time contact dynamics enhancement of space station engineering facility," *IEEE Robotics & Automation Magazine*, pp. 20–28, September 1996.

[7] J. de Carufel, E. Martin, and J.-C. Piedbœuf, "Control strategies for hardware-in-the-loop simulation of flexible space robots," *IEEE Proceedings-D: Control Theory and Applications*, vol. 147, no. 6, pp. 569–579, 2000.

[8] F. Aghili and J.-C. Piedbœuf, "Hardware-in-loop simulation of robots interacting with environment via algebraic differetial equation," in *2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Takamatsu , Japan), pp. 1590–1596, 30 October - 5 November 2000.

[9] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Transactions on Robotics and Automation*, vol. RA-3, no. 1, pp. 43–53, 1987.

[10] J. G. de Jalon and E. Bayo, *Kinematic and Dynamic Simulation of Multibody Systems*. Springer-Verlag, 1989.

[11] M. Doyon, R. L'Archevêque, and J.-C. Piedbœuf, "SPDM task verification facility: Computer architecture and real time implementation," in *DASIA 2000 - Data Systems in Aerospace*, (Montreal, Canada), 22-26 May 2000.

[12] R. L'Archevêque, M. Doyon, J.-C. Piedbœuf, and Y. Gonthier, "SYMOFROS: Sofware architecture and real time issues," in *DASIA 2000 - Data Systems in Aerospace*, (Montreal, Canada), 22-26 May 2000.

[13] D. Rey, J.-C. Piedbœuf, and E. Jackson, "Manipulator design consideration for space robot emulation on ground," in *International Symposium on Robotics and Automotion, ISRA'2000*, (Monterey, Mexico), pp. 41–47, 10-12 November 2000.