7th ESA Workshop on Advanced Space Technologies for Robotics and Automation 'ASTRA 2002'
ESTEC, Noordwijk, The Netherlands, November 19 - 21, 2002

Dynamic Emulation of Space Robot in One-g Environment using Hardware-in-the-Loop Simulation

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

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

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 develop 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 prototype 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 Candarm 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 currently developing 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 Canadian Space Agency ([3],[4]) but also by DLR ([5]) and NASA ([6]).

It is a hardware-in-the-loop simulator consisting in a rigid robot with its control, a simulation of Canadarm 2/SPDM dynamics and a visualisation engine. An operator controls the motion of SPDM through the simulation engine which generates the endpoint motion of the SPDM. That is then used as a setpoint for the robot controller who ensures that the robot endpoint follows the same trajectory as the SPDM. 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. The contact stability problem is similar to the case of force control with a force reflecting master/slave system in teleoperation.

The Canadian Space Agency 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. This paper concentrates on the HLS control law synthesis, on the real-time implementation, and on results from simulation and experience.

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 Candarm 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 currently developing 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 Canadian Space Agency ([3],[4]) but also by DLR ([5]) and NASA ([6]).

In HLS a ground robot is driven by the output of the teleoperated space robot simulation. It is a hardware-in-the-loop simulator consisting in a rigid robot with its control, a simulation of Canadarm 2/SPDM dynamics and a visualisation engine. An operator controls the motion of SPDM through the simulation engine which generates the endpoint motion of the SPDM. That is then used as a setpoint for the robot controller who ensures that the robot endpoint follows the same trajectory as the SPDM. 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. The contact stability problem is similar to the case of force control with a force reflecting master/slave system in teleoperation.

The Canadian Space Agency 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. This paper concentrates on the HLS control law synthesis, on the real-time implementation, and on results from simulation and experience.

2 Controller for Dynamics Emulation

The 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 shall be such that the controlled terrestrial robot is transparent in the frequency band of interest for the analysis required. Two different control approaches can be applied. An intuitive approach consists of using the simulator to replicate the behaviour of the

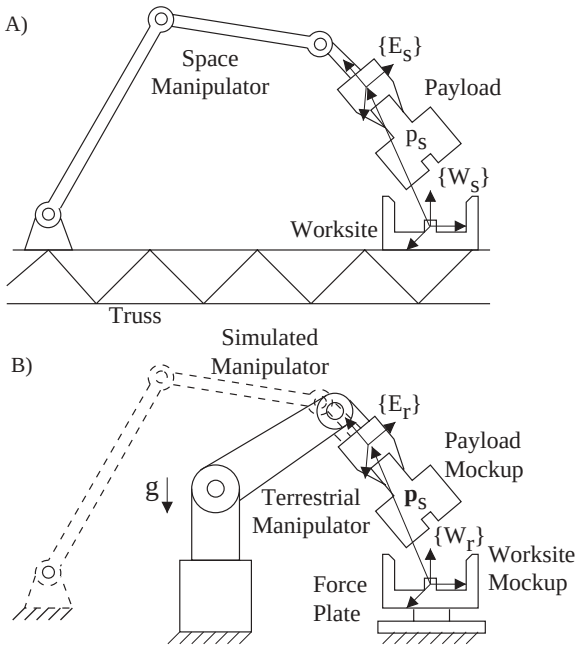

Figure 1: Space Manipulator (A) and Terrestrial Manipulator (B) performing contact task.

reference system subjected to input commands and contact loads, and to feed the terrestrial robot with a reference position to track. However, this approach leads to unstable behaviour in contact. Another approach consists of 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. In this section, we derive this second controller.

2.1 Problem Statement

Figure 1A illustrates a space manipulator handling a payload. The m degree of freedom space robot operates in a n dimensional task space, with $m \geq n \leq 6$. The space robot is typically flexible and redundant. The equation of motion of the space manipulator are described by ([9])

$$\mathbf{M}_s(\hat{\mathbf{q}}_s)\ddot{\hat{\mathbf{q}}}_s = \mathbf{h}_s(\hat{\mathbf{q}}_s, \dot{\hat{\mathbf{q}}}_s, \boldsymbol{\tau}_s) + \mathbf{J}_s^T(\hat{\mathbf{q}}_s)\mathcal{F}_s. \tag{1}$$

In the above $\hat{\mathbf{q}}_s \in \mathbb{R}^m$ is the generalized coordinate vector, $\mathcal{F}_s \in \mathbb{R}^n$ is the vector of force/moment occuring in contact, matrix $\mathbf{M}_s(\hat{\mathbf{q}}_s) \in \mathbb{R}^{m\times m}$ represents the inertia of the manipulator, vector $\mathbf{h}_s(\hat{\mathbf{q}}_s, \dot{\hat{\mathbf{q}}}_s, \boldsymbol{\tau}_s) \in \mathbb{R}^m$ contains Coriolis, centrifugal, friction, stiffness, and the vectors of generalized joint torque $\boldsymbol{\tau}_s$; $\mathbf{J}_s(\hat{\mathbf{q}}_s) \in \mathbb{R}^{n\times m}$ is the Jacobian matrices from the corresponding generalized coordinates to the task space.

We consider the terrestrial robot and the simulated robot, refered by subscripts r and s, with n and m degrees of freedom operating in n dimensional task space. Assume that the mockup of the payload and the actual payload have identical: (i) outside geometry, (ii) local stiffnesses and friction coefficient. For a given object geometry and material properties, we can say:

$$\mathcal{F}_s = \mathcal{F}_r \quad \text{if} \quad \mathbf{p}_s(\hat{\mathbf{q}}_s) \equiv \mathbf{p}_r(\mathbf{q}_r) \tag{2}$$

where $\mathbf{p}_s \in \mathbb{R}^6$ is the pose of the origin of the end effector frame $\{E_s\}$ with respect to the fixed reference frame attached to the worksite $\{W_s\}$, and vector $\mathbf{q}_r \in \mathbb{R}^n$ represents the joint angles of the terrestrial robot. The inference in (2) suggests that the terrestrial robot produces the same contact force as the space robot does, if the payload mockup has the same relative motion as what would be encountered in space.

2.2 Control

The control is realized by constraining the endpoint motion of the terrestrial robot to the one of a real-time simulator representing the dynamics of the space robot in free-space, as illustrated in Fig. 1B. The terrestrial robot must be controlled such that the contact force equals the constraint force resulting from the kinematic constraint. Therefore, the hardware-in-loop simulation system obtained by combining the simulator and the terrestrial manipulator, is virtually equivalent to the original space robot in Fig. 1A.

The equation of motion of the terrestrial robot interacting with the environment is described by ([9])

$$\mathbf{M}_r(\mathbf{q}_r)\ddot{\mathbf{q}}_r = \bar{\mathbf{h}}_r(\mathbf{q}_r, \dot{\mathbf{q}}_r) + \boldsymbol{\tau}_r + \mathbf{J}_r^T(\mathbf{q}_r)\mathcal{F}_r, \tag{3}$$

where the robot parameters $\mathbf{M}_r$, $\mathbf{h}_r = \bar{\mathbf{h}}_r(\mathbf{q}_r, \dot{\mathbf{q}}_r) + \boldsymbol{\tau}_r$, and $\mathbf{J}_r$ are defined as previously for the space robot (1). We assume that a simulator capture the dynamics of the space robot in a free space by the following equation of motion

$$\mathbf{M}_s(\mathbf{q}_s)\ddot{\mathbf{q}}_s = \mathbf{h}_s(\mathbf{q}_s, \dot{\mathbf{q}}_s, \boldsymbol{\tau}_s). \tag{4}$$

As mention in Eq. 2, it is imperative to equate the endpoint pose of the two manipulators. This requirement is formally expressed by a kinematic constraint as

$$\boldsymbol{\Phi}(\mathbf{q}_r, \mathbf{q}_s) = \mathbf{p}_r(\mathbf{q}_r) - \mathbf{p}_s(\mathbf{q}_s) \equiv 0, \tag{5}$$

which is a system of n equations.

We apply this constraint to the augmented system obtained by combining the simulated space robot (4) with the model of the terrestrial robot (3) to obtain a set of $2n + m$ equations

$$\mathbf{M}_s\ddot{\mathbf{q}}_s + \boldsymbol{\Phi}_{\mathbf{q}_s}^T\boldsymbol{\lambda} = \mathbf{h}_s \tag{6}$$

$$\mathbf{M}_r\ddot{\mathbf{q}}_r + \boldsymbol{\Phi}_{\mathbf{q}_r}^T\boldsymbol{\lambda} = \bar{\mathbf{h}}_r(\mathbf{q}_r, \dot{\mathbf{q}}_r) + \boldsymbol{\tau}_r + \mathbf{J}_r^T(\mathbf{q}_r)\mathcal{F}_r \tag{7}$$

$$\boldsymbol{\Phi}(\mathbf{q}_r, \mathbf{q}_s) = 0 \tag{8}$$

where $\boldsymbol{\Phi}_{\mathbf{q}_s} = \partial\boldsymbol{\Phi}/\partial\mathbf{q}_s = -\mathbf{J}_s(\mathbf{q}_s) \in \mathbb{R}^{m\times n}$ denotes the Jacobian of the constraint with respect to $\mathbf{q}_s$, $\boldsymbol{\Phi}_{\mathbf{q}_r} = \partial\boldsymbol{\Phi}/\partial\mathbf{q}_r = \mathbf{J}_r(\mathbf{q}_r) \in \mathbb{R}^{n\times n}$ denotes the Jacobian of the constraint with respect to $\mathbf{q}_r$, and $\boldsymbol{\lambda} \in \mathbb{R}^n$ represents the Lagrangian multiplier, that is the force necessary to maintain the constraint condition. Equations (6), (7) and (8) are a system of *Differential Algebraic Equation* (DAE) that completely formulates the constrained simulator.

A comparison of equation 6 (with $\boldsymbol{\Phi}_{\mathbf{q}_s} = -\mathbf{J}_s(\mathbf{q}_s)$) and the target dynamics (1), reveals that the Lagrangian multiplier, $\boldsymbol{\lambda}$, acts as a contact force/moment to the simulated robot. Therefore, our goal is to obtain the torque that we must apply to the terrestrial robot such that the constraint force, $\boldsymbol{\lambda}$, equates the contact force, $\mathcal{F}_r$. As done typically in Lagrange Multiplier approach, we differentiate the constraint equation twice to be able to solve first for the acceleration and then for the constraint.

$$\boldsymbol{\Phi}_q\ddot{\mathbf{q}} + \dot{\boldsymbol{\Phi}}_q\dot{\mathbf{q}} = 0, \tag{9}$$

where $\boldsymbol{\Phi}_q = \begin{bmatrix} \mathbf{J}_r & -\mathbf{J}_s \end{bmatrix} \in \mathbb{R}^{(n+m)\times n}$ denotes the Jacobian of the constraint with respect to the augmented coordinate vector obtained by combining $\mathbf{q}_r$ and $\mathbf{q}_s$. We demonstrated in [8] that the solution of (6), (7) along with (9) with $\boldsymbol{\lambda} = \mathcal{F}_r$ results in the following expression for $\boldsymbol{\tau}_r$:

$$\boldsymbol{\tau}_r = -\bar{\mathbf{h}}_r - \mathbf{M}_r\mathbf{J}_r^{-1}\dot{\mathbf{J}}_r\dot{\mathbf{q}}_r - \mathbf{J}_r^T\mathcal{F}_r + \mathbf{M}_r\mathbf{J}_r^{-1}\hat{\mathbf{a}}_s \tag{10}$$

Where

$$\hat{\mathbf{a}}_s = \mathbf{J}_s\mathbf{M}_s^{-1}\mathbf{h}_s + \dot{\mathbf{J}}_s\dot{\mathbf{q}}_s + \mathbf{J}_s\mathbf{M}_s^{-1}\mathbf{J}_s^T\mathcal{F}_r \tag{11}$$

is the Cartesian tip acceleration of the reference model robot calculated from the unconstrained dynamical model. We have shown in [8] that the control law in (10) linearized the robot prototype so that it acts virtually as a double integrator. However, the linearized robot inevitably exhibits drift. We improve the control law by incorporating feedback loops on the drift error, which play a further important role in attenuating the effect of disturbance and model uncertainty. The new control law with feedback is

$$\boldsymbol{\tau}_r' = -\bar{\mathbf{h}}_r - \mathbf{M}_r\mathbf{J}_r^{-1}\dot{\mathbf{J}}_r\dot{\mathbf{q}}_r - \mathbf{J}_r^T\mathcal{F}_r + \mathbf{M}_r\mathbf{J}_r^{-1}\left(\hat{\mathbf{a}}_s - \mathbf{G}_v(\mathbf{J}_r\dot{\mathbf{q}}_r - \int\hat{\mathbf{a}}_s dt) - \mathbf{G}_p(\mathbf{p}_r(\mathbf{q}_r) - \int\int\hat{\mathbf{a}}_s dt)\right) \tag{12}$$

where $\mathbf{G}_v, \mathbf{G}_p \in \mathbb{R}^{n\times n}$ are drift compensation gains.

The controller is a model based cartesian linearisation with acceleration input. To avoid the inevitable drift, we added a PD control on the cartesian position and velocity. This is a complete analogy to the Baumgarte stabilisation used in constraint dynamical systems. Figure 2 illustrates this controller.

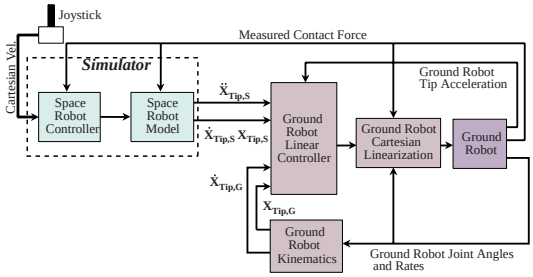

Figure 2: Cartesian Feedback Linearization with Acceleration Controller

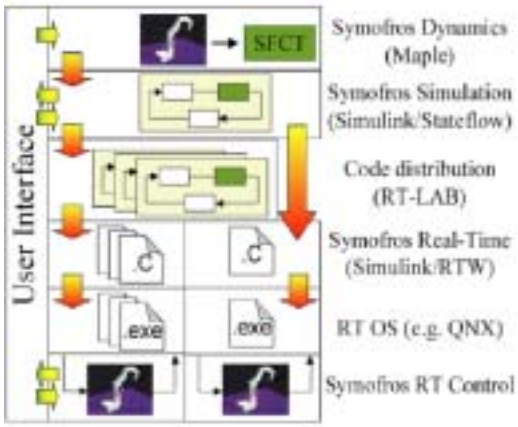

Figure 3: Symofros Architecture

3 Software Development and Implementation

The system development and implementation is done using Symofros environment ([10],[11]). As shown in Fig. 3, Symofros is used from the development stage up to the real-time implementation phase of the project. The main guideline was to exclude any hand recoding to go from the simulation used in the development to the real-time code used in the control of the terrestrial robot.

In a typical project, we first need to do some quick modelling to assist in the design phase. This is done using the Symofros graphical user interface and the symbolic dynamic model generator based on Maple. Using these models, we can refine the specifications and make the final design of the robot. While the robot is being build, we can start the controller development. This step has involved many iterations to find the right controller design since the hardware-in-the-loop control is still a recent research area. This step is done in a non real-time simulation under Simulink using a library of model based functions resulting from the symbolic model generator.

When a good controller is obtained, we need to test it in real-time simulation. In this simulation, we reproduce the exact inputs and outputs with the same quantisations and the same delays. This is important to have a behaviour similar to the real system. The goal is to be able to replace the model of the robot by the real robot for the implementation. No code is modified between the simulation, the real-time simulation and the real-time implementation.

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 unique 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.

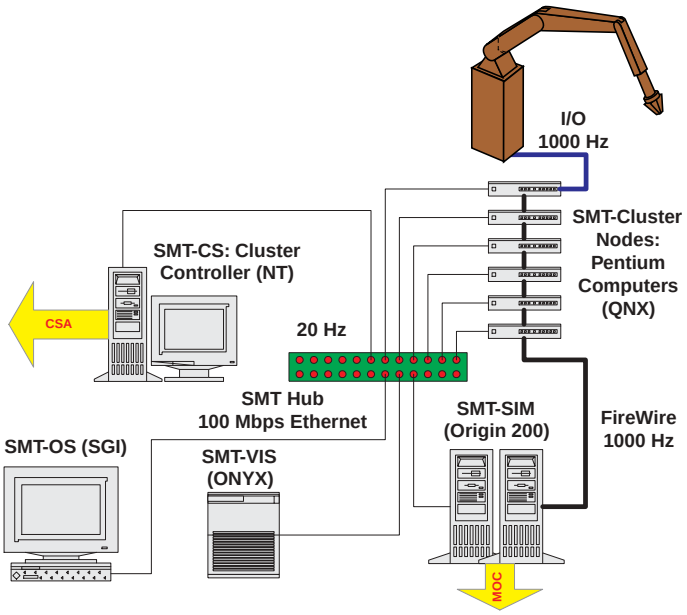

Figure 4: Computer Architecture for the Space Robot Dynamics Emulation

3.2 Implementation

We are using a heterogeneous computer environment to maximize the use of existing Canadian Space Agency simulation facilities (see Fig. 4). The controller of the terrestrial robot is running on a cluster of six Pentium under QNX. The real-time simulator of the space robot used the MSS Operation and Training Simulator (MOTS). The dynamic engine of MOTS (SMT-SIM) is running on a SGI Origin 200 machine with four processors while the visual engine (SMT-VIS) 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 onsystem architecture hardware and software see [10].

The alllocation 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 change 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) together with the major system state machines. This node has two firewire cards: one for the communication with other computers of the cluster and one for the communication with space robot simulator (SMT-Sim). 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 cluster is directly manage 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

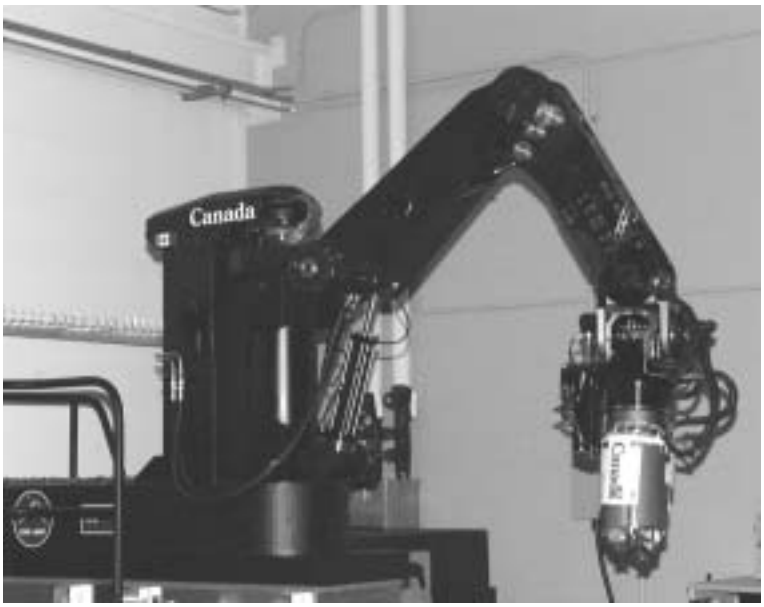

Figure 5: STVF Testbed Manipulator

interface to third party GUIs, e.g. Labview or custom designed GUIs.

4 Simulation and Experimental Results

Since an off-the-shelf industrial manipulator could not meet our requirements, a custom hydraulic manipulator (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 [12].

The peg on the plane was also tested experimentally. Figure 6 shows the velocity of the simulated SPDM and of the terrestrial robot (SMT). Figure 7 shows the resulting experimental forces measured in three directions.

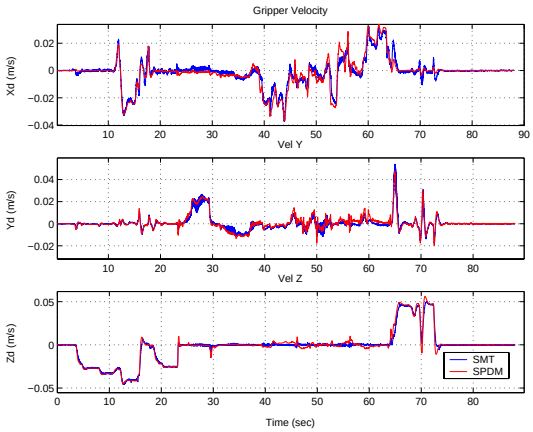

Figure 6: Experimental Velocities for Peg on Plane

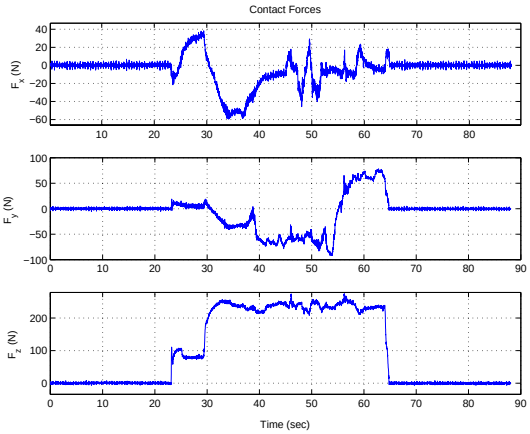

Figure 7: Experimental Forces for Peg on Plane

Initially, the robot is moving in free space. Then, after 23 s, the end effector is moving toward the plane in the z direction until the peg hit the plane. The force is first stabilised around 100 N and then increased above 200 N while the peg is moving in x and y direction. The forces F_x and F_y represent the friction forces on the plane. It can be seen on Figure 6 that the terrestrial robot (SMT) is tracking the space robot (SPDM) with a good accuracy. This prove that the HLS control scheme proposed compensated the dynamics of the terrestrial robot.

The last set of experimental results, presented in Figures 8, 9 and 10, shows a peg-in-hole insertion. Here the hole is mounted on a rigid force plate to accurately measure contact forces and moments. This means that we have here a very stiff robot contacting a rigid worksite, controlled in such a way to replicate the dynamics of a flexible system. Here, the peg is positioned a few centimeters above the hole with a small misalignment. The SPDM is commanded to move at 4 mm/s in the axial direction. After about 16 s, the peg hits the side of the hole and the force moment accommodation scheme of the SPDM realigns the peg to permit the insertion. The insertion is done at 4 mm/s and the speed is reducing to 2 mm/s just before hitting the bottom of the hole. When the peg hits the bottom of the hole, the Z-force stabilizes to 50 N, as seen in Figure 10. This force is increased to 100 N by commanding the SPDM to move at a velocity of 4 mm/s. Although these velocities and forces seem low, they are typical of operational velocities of the SPDM in space. From Figure 8, it is observed that the error in the Y and Z position of the SMT robot is below our maximum allowable error of 5 mm. However, the error for the X-axis attains 1 cm. This error will be reduced by completing the dynamics identification of the robot since it is due to the linearization errors in our cartesian feedback linearisation scheme. Finally, from Figure 9, we observe some undesired vibrations of the SMT robot when reacting to a change on desired velocity. This problem will be solved by adjusting its controller gains. Nevertheless, these results are quite impressive since we can achieve stable contact operations with this HLS control scheme while already having reasonnable performances.

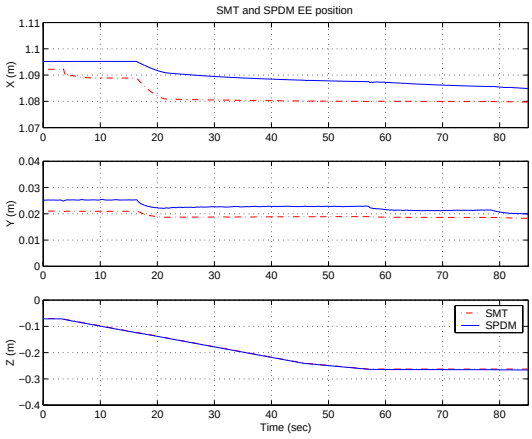

Figure 8: Experimental Positions for Peg in Hole

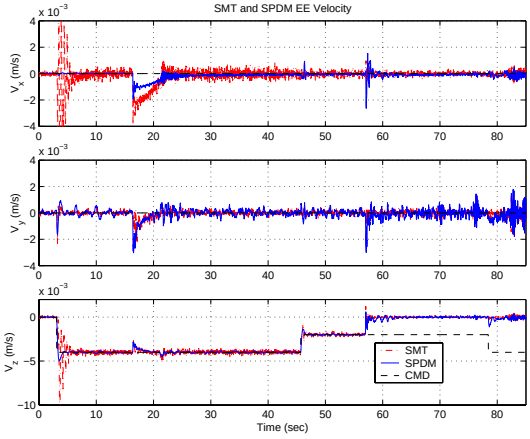

Figure 9: Experimental Velocities for Peg in Hole

5 Conclusion

In this paper, we demonstrated that a space robot can be emulated by a terrestrial manipulator using a hardware-in-the-loop simulation technique. We shown that the controller need to cancel the dynamics of the terrestrial robot. This is done through a cartesian feedback linearisation technique with the acceleration of the space robot endpoint as input. We also discussed the critical aspect of the implementation of such a controller on a real robot. We have used Symofros, a software allowing quick prototyping and real-time implementation. We demonstrated the validity of our approach in simulation and experimentally using a peg on the plane and experimentally using the more complex peg-in-hole test case. In both cases, we proved that the proposed hardware-in-the-loop concept can emulate a contact task of a space robot.

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.

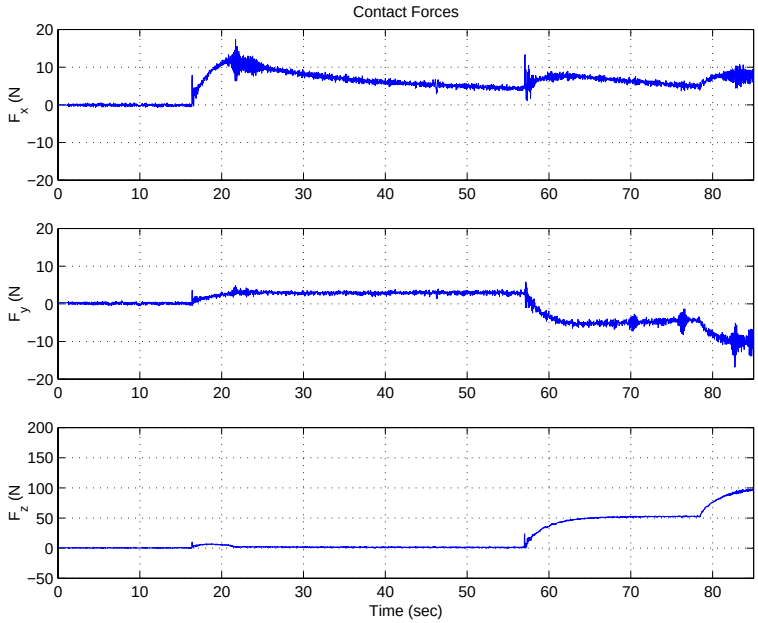

Figure 10: Experimental Forces for Peg in Hole

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

[10] 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.

[11] 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.

[12] 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.