

Real-Time Simulation of a Space Robotic Arm

Gianni Ferretti, *Member, IEEE*, Gianantonio Magnani, *Senior member, IEEE*, Paolo Porrati, Gianpaolo Rizzi, Paolo Rocco, *Member, IEEE*, and Andrea Rusconi

Abstract—The paper presents the DEXARM Real-Time Simulation (DRTS) tool conceived to support the design and development of the controller of the Dexterous Robot Arm (DEXARM), a lightweight 7-d.o.f. space robotic arm currently under development. The assembly of the robot model relies on existing Modelica model libraries, whereas the controller can be described as a regular Matlab/Simulink model. Model and controller tasks run under Linux-RTAI real-time kernel, together with a third task providing interface functions and animation of a 3D geometric and kinematic model of the arm. A novelty of the paper concerns the simulation of friction: the adoption of a closed form solution of the friction model equations allows simulating both sliding and pre-sliding regimes with a fixed-step algorithm.

I. INTRODUCTION

REAL-time simulation systems are mainly used for testing and check out of control electronics and other components of complex systems (hardware-in-the-loop simulation), like power plants, aircrafts, vehicles, as well as for training of plant operators, aircraft pilots, and astronauts.

In real-time simulators, the inputs to the model are obtained from external devices each sampling time. Model equations must then be solved within prescribed time intervals: this way a selected subset of variables can be computed and presented as outputs at the next sampling time. In order to implement real-time communication with external world and to schedule model execution exactly each sampling time, the simulation software relies on real-time kernels, through calls to system primitives included in the numerical solution code. Usually, an effort is also necessary to simplify those model equations that entail a significant computational burden.

Commercial tools exist that adapt offline models to real-time simulations on dedicated hardware. A typical situation consists in porting Simulink models to dedicated hardware

using the Matlab Real-Time Workshop [1] or the dSPACE TargetLink [2].

Simulink as well as Dymola [3] models can be interfaced to dSPACE hardware in order to execute hardware in the loop simulations. Real-time tasks can be also derived [4] from a standard Modelica model compiled in Dymola. Simulink based simulators are described in [5], [6], whereas the simulator proposed in [7] is based on VR software.

There is also research on porting the simulators obtained with open-source modelling tools like MBDyn [8] onto real-time, possibly distributed, platforms, like RTnet [9]. Generation of parallel code from Modelica models is also investigated [10].

This paper contributes to this research on real-time simulation of mechanical system presenting the DEXARM Real Time Simulator (DRTS), a tool conceived and realized as an aid to the design of the controller of a new space robotic arm. DEXARM is a 7-d.o.f. lightweight space manipulator arm currently under development on behalf of the European Space Agency (ESA).

DRTS has several distinctive features:

- The real-time simulation code is obtained directly from offline developed Modelica models. All the powerful Modelica libraries as well as the features and tools of a Modelica editor/compiler, like Dymola, can be exploited for the development of new models;
- The controller model can be either a regular Matlab/Simulink model (internal controller), or a special purpose hardware software controller to be tuned and tested (external controller);
- A detailed and validated model of the arm joint is included, featuring friction model, torsional flexibility, and torque ripple in the brushless motors. In particular, sliding and pre-sliding regimes of friction have been modeled, even if a fixed-step numerical integration algorithm has to be used, to comply with the real time simulation constraint. This result has been obtained thanks to the adoption of the Single State Integral Friction Model (SSIFM) [11]. SSIFM gives results that are very close to the output of the LuGre model [12], adopting however a closed form solution of the differential and discontinuous friction model equations.
- The interactive human-machine interface supports the motion command input, the introduction of force disturbances through a joystick, and the 3D visualization of the arm motion;
- The simulator tasks run under the Linux operating

Manuscript received June 27, 2008. This work was supported in part by the Lombardy Region within the ICT Metadistrict Project Si Parte!

G. Ferretti is with Politecnico di Milano, Milano, Italy (phone: +39 02-23993682; fax: +39 02-23993412; e-mail: ferretti@elet.polimi.it).

G. Magnani is with Politecnico di Milano, Milano, Italy (phone: +39 02-23993673; fax: +39 02-23993412; e-mail: magnani@elet.polimi.it).

P. Porrati is a former student at Politecnico di Milano, Milano, Italy

G. Rizzi is with IIS Jean Monnet, Mariano Comense, Italy (e-mail: gprizzi@tin.it)

P. Rocco is with Politecnico di Milano, Milano, Italy (phone: +39 02-23993685; fax: +39 02-23993412; (e-mail: rocco@elet.polimi.it).

A. Rusconi is with Selex Galileo, Via Montefeltro 8, 20156 Milano, Italy (e-mail: andrea.rusconi@selexgalileo.com).

within an offline simulation model. The DEXARM model includes descriptions of the dynamics of the current control electronics, the motors, and the gearboxes, as well as of the arm mechanical chain model.

The 3D visualization of the robot motion, the handling of the joystick, and the interactive input of motion commands are the functions implemented by the human-machine interface.

To keep the picture readable, the offline data analysis functions are not shown in Fig. 2. Actually the user of the simulator is allowed to select a set of model variables to be recorded for an offline analysis. Data are saved by the DEXARM model task during simulation and can be easily analyzed, once the real-time session has been terminated.

IV. SOFTWARE ARCHITECTURE

The real-time software runs under the Linux operating system with the *Real Time Application Interface* (RTAI) extension. RTAI is a hard real time extension of Linux, which makes it capable of handling time critical tasks in a predictable way. It provides task scheduling and synchronization, and inter-task communication services, among others.

The DRTS environment (see Fig. 3) consists of three major Linux-RTAI tasks: the CONTROLLER, the MODEL, and the HMI tasks. The CONTROLLER is a periodic task scheduled according to the servo-loop sampling time (either 0.4 or 1 ms) that runs in kernel space. The MODEL and the HMI tasks run in user space under the RTAI module LXRT.

LXRT [16] supports the development and execution of soft and hard real-time in user space, giving access to both Linux (for instance to operate the TCP/IP communications) and RTAI services symmetrically. LXRT hard real-time in user space allows full kernel pre-emption, with the only penalty of a very slight increase in overhead, whereas jitter and latency remain very close to those measurable for the same applications implemented in kernel space. The only constraint is that Linux kernel services cannot be used directly. To access RTAI services from LXRT, a Linux process has to create a real-time kernel task, called the *buddy/proxy*. This task is in charge of the execution of real-time services, in particular inter-task mailbox based communication services. While the execution in user space is rather natural for the low priority HMI task, for the highly demanding, in terms of CPU time, MODEL task it is motivated by the aim of avoiding critical CPU blocks.

Inter-task communication is obtained using the remote mailboxes provided by RTAI. In this way the controller and the simulator may run on different machines, if necessary. As no direct communication between tasks in user space and in kernel space is allowed, the communication between the MODEL and CONTROLLER tasks, as well as between the HMI and the CONTROLLER tasks, occurs through buddy/proxies, created by a proper LXRT function, that directly handle the communication mailboxes.

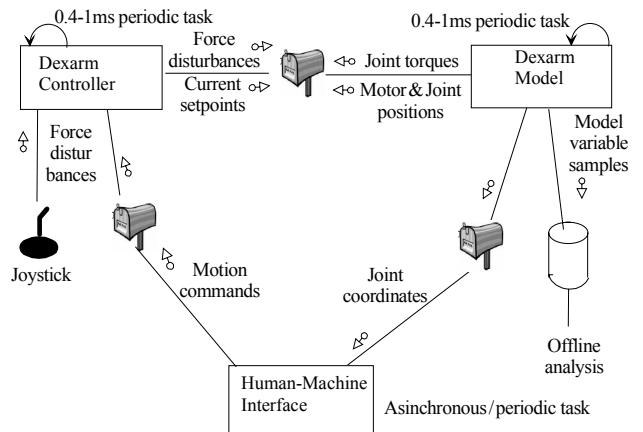


Fig. 3 DRTS software architecture

A. Task Scheduling and Synchronization

While the CONTROLLER task execution is scheduled periodically by a RTAI primitive, the synchronous execution of MODEL with CONTROLLER is achieved by a blocking inter-task communication mailbox. The MODEL executes, and advances the simulation time of a single sampling period, only after the CONTROLLER task has written the new set of current setpoints (the inputs to the model).

The HMI task implements the periodically executed 3D animation function and asynchronous input functions supporting the input of motion commands. Further details on the HMI operations are given in the specific Section.

B. CONTROLLER Task

The CONTROLLER task is obtained from a regular Simulink model, developed in a Matlab environment running either under Linux or MS-Windows, and converted into C programs by the Real Time Workshop Matlab toolbox.

Relying on the approach and software tools provided by RTAI-Lab ([17], [18]), a special Simulink block (actually an S-function) has been developed. Once included within the model, this block provides the inter-task communication required to match the hard real-time execution of the CONTROLLER with the execution of the Modelica model. A similar second block implements the communication with the HMI task.

The CONTROLLER task handles the joystick interface too: being a real-time high priority task with high scheduling frequency it ensures a fast and predictable response to the joystick commands. Nevertheless the more natural alternative of handling the joystick through the HMI task is under evaluation at present.

The CONTROLLER task is also used to interface a physical external controller. In this case it reduces to a pure interface task.

The real time execution of the RTW model can be slowed down (or time scaled) if needed, for debugging purposes.

C. MODEL Task

The arm model is developed offline within the standard Dymola environment. A special purpose block (ModRTAI, Modelica code which relies on a library of C subroutines [4]) has been created. Once included in the model, this block creates the buddy/proxy, selects the input and output model variables, and implements the communication and synchronization functions by invoking the proper RTAI primitives. In this way, the C file generated by the compilation of the model can easily become a RTAI executable task. At every wakeup, it reads the inputs from the CONTROLLER task, evaluates the new model state and outputs, and writes the latter to the output mailboxes. Similar to the CONTROLLER task, ModRTAI supports the execution of the DEXARM model at a speed slower than the real time one.

As far as the numerical solution algorithm of the model is concerned, the Dymola Inline Integration method, applied to the implicit Euler algorithm, has been selected in order to obtain the fastest simulation speed.

The DRTS is currently running in real time in a dual core PC with the operating system Kubuntu Linux 7.10, kernel 2.6.22-15, and RTAI 3.6.

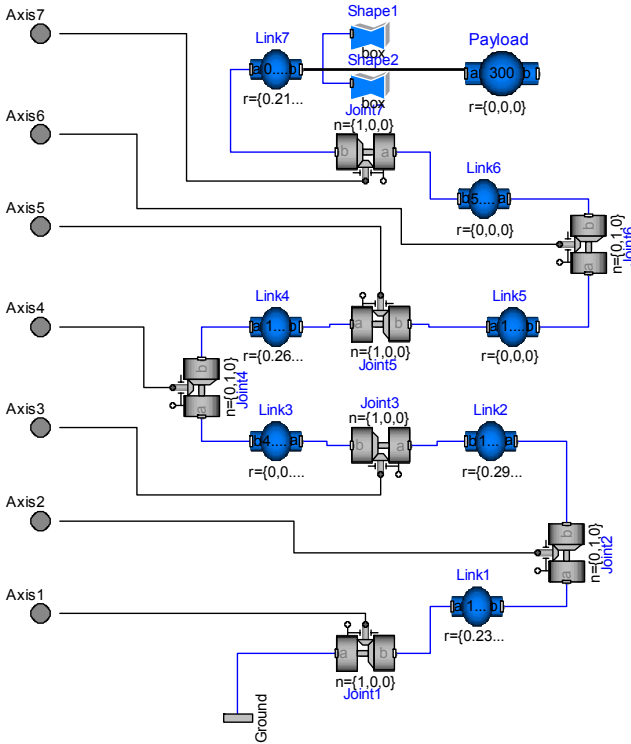


Fig. 4 Mechanical chain model

V. DEXARM MODEL AND CONTROLLER

The DEXARM model comprises the arm mechanical chain, the joints, the motors, and the current control electronics. The Dymola implementation consists of two

major components: the Servo model, instantiated seven times, and the Mechanical chain model. The latter (see Fig. 4) is built using the “ActuatedRevolute” model (seven instantiations) and the “BodyShape” (eight instantiations) of the Modelica Mechanics MultiBody Library.

The Servo model includes the current servoloop, the three phase AC brushless motor, and harmonic drive gear models.

A. Current servoloop model

This component models the field oriented control of the motor currents. The current sensor biases have been simulated, as they are responsible of torque pulsation harmonics (ripple) which are visibly detrimental for the quality of the motion of the arm.

B. AC brushless motor model

The torque generation in a brushless motor is modelled as shown in Fig. 5. For each phase of the motor the current and the back-emf are computed, which jointly contribute to generate the torque. The nominal torque is given by $\tau_m = K_t I_{ref}$, where K_t is a constant characteristic of the motor and I_{ref} the setpoint of the amplitude of each phase current. A fast enough current controller is assumed. However, higher order harmonics in both the current and the back-emf shapes cause additional undesired torque pulsations [18], in turn causing visible oscillations of the arm, especially at low and intermediate velocities. Non-sinusoidal back-emf profiles have been therefore simulated, and the oscillations obtained from experiments have been predicted by the model [20].

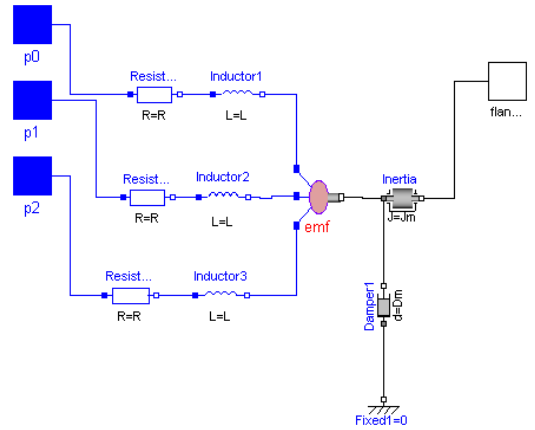


Fig. 5 Brushless motor model

C. Harmonic drive model

Transmission chains and gearboxes are major sources of friction and torsional flexibility in position servomechanisms. Friction non linear effects and discontinuities manifest and dominate the arm behavior at very low speed and especially around velocity zero crossing. As these conditions frequently occur in compliant motion, special care has been given to friction modeling and simulation. Several friction models have been checked, with respect to modeling accuracy and computational burden,

Higher level control functions (e.g., kinematic inversion, trajectory generation) can easily be added to the servo level controller to simulate operational space control operation. Finally, the CONTROLLER task can be used to interface the real arm control unit to the simulation model and HMI tasks.

VI. HUMAN-MACHINE INTERFACE

The human-machine interface supports the definition of the arm 3D model, the 3D real time animation of such model according to the computed robot motion, and the interactive assignments of trajectories and of their via points. Fig. 10 shows a snapshot of the interface, where the drawing tools are visible and a 3D model of the whole arm has been composed. This 3D model can either be obtained using a built-in CAD tool or it can be imported from an Open Inventor [23] or a Virtual Reality Model Language (VRML) model. Using the link models, the kinematic structure of the arm is defined interactively to be used for 3D animation of the arm motion and interaction with the environment.

Animation is driven by the joint coordinates read either from file, as it is in the offline simulation, or from a mailbox in real time simulation. The software architecture of the HMI is shown in Fig. 11.

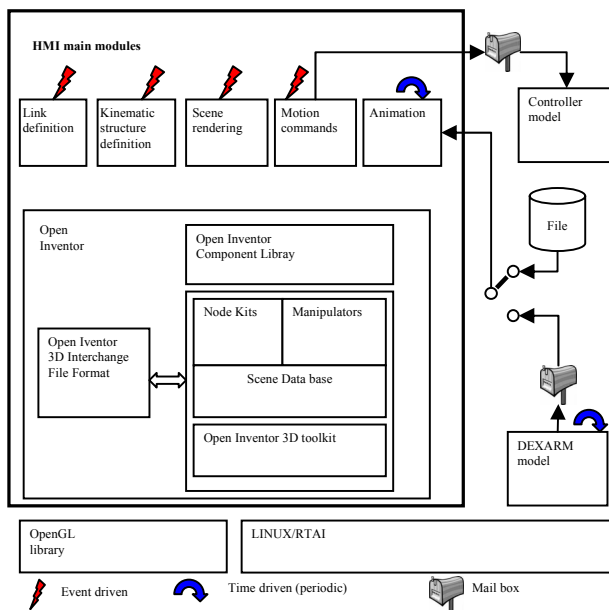


Fig. 11 HMI major software modules and external interfaces

The Open Inventor package, released by Silicon Graphics as open source software, is based on the objects/events paradigm. The user describes the objects in a 3D scene as a graph, defines a set of events and writes the call back routines associated to each event. The event handler of the library will start the execution of the corresponding call back routine whenever an event occurs. Examples of events are the selection of an object by the mouse, the displacement of an object, or an object reaching a predefined position. In addition to the asynchronous events, the library supports

periodic events that are generated repeatedly based on a specified time period.

A periodic event is used to periodically update the 3D model of the arm, according to the joint coordinates computed by the simulator. The event triggers the execution of the associated call back function which reads from the input mailbox the last set of joint coordinates written by the simulator and updates the matrices defining positions and orientations of the arm links. This in turn generates an asynchronous event which, through the event handler, starts the Open Inventor functions which update the 3D scene.

Thus the scene rendering and animation process is not strictly real time, and few frames may also go lost, but without degrading the visual feedback.

The fast graphic rendering capability is obtained by the OpenGL functions, which exploit the performance of available, even low cost, graphic accelerators.

VII. CONCLUSIONS

An approach has been proposed to develop real-time simulators of complex electromechanical systems by exploiting the most powerful non real-time modeling and control design tools. An open source general purpose operating system like Linux is used, still assuring the satisfaction of the most critical time deadlines thanks to the RTAI extension.

This approach relies on standard and commercial tools and on open source packages, and required the development of few interface blocks to be included within the Simulink and Dymola models, respectively. The modeling and validation work carried out on a joint prototype in the early phase of the arm development process could be fully included in the real-time simulation model, achieving quite accurate and reliable results almost effortlessly.

The Simulink arm controller description can also be easily tested in an incremental way. At the end the real DEXARM controller can be interfaced to drive the arm real-time model for tuning, testing and checkout purpose.

A significant effort has been devoted to create a human machine interface able to support the input of motion commands and force disturbances, together with the 3D visualization of the arm motion, relying on a powerful open source package.

ACKNOWLEDGEMENT

Contributions are acknowledged from William Spinelli, to the software architecture design and development, and from Luca Viganò and Dario Camorali, to the Dexarm model development.

REFERENCES

- [1] Real-Time Workshop: Generate C code from Simulink models and MATLAB code [Online]. Available: <http://www.mathworks.com/products/rtw/>
- [2] dSPACE - Solutions for Control [Online]. Available: <http://www.dspaceinc.com/>

- [3] Dymola Multi-Engineering Modeling and Simulation [Online]. Available: <http://www.dynasim.se/>
- [4] G. Ferretti, M. Gritti, G. Magnani, G. Rizzi and P. Rocco, "Real-time simulation of Modelica models under Linux/RTAI," *Proc. 4th Int. Modelica Conf.*, 2005, pp. 359–365.
- [5] S. S. Ge, T.H. Lee, D.L. Gu, and L.C. Woon, "OpenRob: An open_architecture platform for model building, controller design and numerical simulation", *IEEE Rob. Aut. Mag.*, Vol. 7, No. 3, pp. 42–54, 2000
- [6] W.E. Dixon, D. Moses, I. D. Walker, D. M. Dawson, "A Simulink-based robotic toolkit for simulation and control of the Puma 560 robot manipulator," *Proc. Int. Conf. Intell. Rob. Syst.*, 2001, pp. 2202–2207
- [7] F. Isnard, G Dodds, C Vallée, and D Fortuné, "Real-time dynamics simulation of a closed-chain robot within a virtual reality environment", *Proc. Inst. Mech Eng., Part K*, Vol. 214, No. 4, pp. 219–232, 2000.
- [8] M. Attolico and P. Masarati, "A multibody user-space hard real-time environment for the simulation of space robots," *Proc. Real-Time Linux Workshop*, 2003.
- [9] RTnet – Hard real-time networking for Linux/RTAI [Online]. Available: <http://www.rts.uni-hannover.de/rtnet/>
- [10] P. Aronsson, P. Fritzson, "Multiprocessor scheduling of simulation code from Modelica models," *Proc. 2nd Int. Modelica Conf.*, 2002, pp. 331–338.
- [11] G. Ferretti, G. Magnani, P. Rocco, "An Integral Friction Model," *Proc. Int. Conf. Rob. Autom.*, 2004, pp. 1809 – 1813.
- [12] C. C. de Wit, H. Olsson, K. J. Åström and P. Lischinsky, "A new model for control of systems with friction," *IEEE Trans. Aut. Contr.*, Vol. 40, No. 3, pp. 419–425, 1995.
- [13] D. Beal, E. Bianchi, L. Dozio, S. Hughes, P. Mantegazza, and S. Papacharalambous, "RTAI: Real time application interface," *Linux Journal*, April 2000.
- [14] A. Rusconi, P. Magnani, T. Grasso, G. Rossi, J.F. Gonzalez Lodoso and G. Magnani, "DEXARM - a dexterous robot arm for space applications," *Proc. ASTRA 2004 Workshop*, 2004.
- [15] A. Rusconi, P. Magnani, J.F. Gonzalez Lodoso, P. Campo, R. Chomicz and G. Magnani, "Design and development of an integrated joint for the dexterous robot arm," *Proc. ASTRA 2004 Workshop*, 2006
- [16] G. Quaranta and P. Mantegazza, "Using Matlab-Simulink RTW to build real time control applications in user space with RTAI-LXRT," *Proc. Real Time Linux Workshop*, 2001
- [17] R. Bucher and L. Dozio, "CACSD with Linux RTAI and RTAI-Lab," in *Proc. Real Time Linux Workshop*, 2003.
- [18] E. Bianchi and L. Dozio, "Some experience in fast hard real-time control in user space with RTAI-LXRT," *Proc. Real Time Linux Workshop*, 2000.
- [19] G. Ferretti, G. Magnani and P. Rocco, "Modelling, identification and compensation of pulsating torque in permanent magnet AC motors", *IEEE Trans. on Ind. Electr.*, Vol. 45, No. 6, pp. 912-920, 1998.
- [20] G. Magnani, P. Rocco and A. Rusconi, "Modeling and position control of a joint prototype of DEXARM", *Proc. Int. Work. Adv. Mot. Contr.*, 2008, pp. 562-567
- [21] H.D. Taghirad and P. R. Belanger, "An experimental study on modelling and identification of harmonic drive systems," *Proc. Int. Conf. Dec. Contr.*, 1996, pp. 4725 – 4730
- [22] M. Spong, "Modeling and control of elastic joint robots", *ASME Journ. Dyn. Syst., Meas. Contr.*, Vol. 109, pp. 310-319, 1987.
- [23] Open Inventor [Online]. Available: <http://www.oss.sgi.com/projects/inventor/>