MULTIBODY DYNAMICS 2005, ECCOMAS Thematic Conference
J.M. Goicolea, J. Cuadrado, J.C. García Orden (eds.)
Madrid, Spain, 21–24 June 2005

A REAL-TIME HARDWARE-IN-THE-LOOP SIMULATOR FOR ROBOTICS APPLICATIONS

Marco Morandini[⋆]**, Pierangelo Masarati**[⋆]**, and Paolo Mantegazza**[⋆]

[⋆]Dipartimento di Ingegneria Aerospaziale
Politecnico di Milano
Campus Bovisa, via La Masa 34, 20156 Milano, Italy
e-mail: marco.morandini@polimi.it, pierangelo.masarati@polimi.it,
paolo.mantegazza@polimi.it,
web page: http://www.aero.polimi.it/

Keywords: Real-Time Simulation, Robotics, Computational Mechanics, Open-Source Software.

Abstract. *This paper presents an application to robotics modeling of the multibody free software MBDyn in its real-time variant. Real-time simulation of robots is usually carried on using dedicated software because of the stringent requirements in terms of solution time for each time step. The real-time variant of the general-purpose multibody simulation software MBDyn originated from the need to share a common simulation environment for the detailed and the real-time analysis of complex, heterogeneous multidisciplinary mechanisms, like robots, helicopter and tiltrotor wind-tunnel models, aircraft landing gear and hydro-mechanical components, and more.*

The capability to simulate the same problem with increasingly detailed models, ranging from compact models for real-time applications to highly sophisticated models for fully detailed analysis and verification, sharing the same model layout and common parts where limited detail suffices, represents a great advantage because the analyst can work in a single, dependable environment.

This paper illustrates recent advances in the real-time performances of the simulation framework based on MBDyn for space robotics control applications; focus is on performances improvement when addressing "small" and "very small" (in a sparse matrix sense) problems, and on realistic joint friction models.

1 INTRODUCTION

This paper presents an application to robotics modeling of the multibody free software MBDyn (`http://www.aero.polimi.it/~mbdyn/`) in its real-time variant. Real-time simulation of robots is usually carried on using dedicated software because of the stringent requirements in terms of solution time for each time step. Recently, the need to perform real-time simulation of arbitrary mechanisms, ranging from industrial robots to rotorcraft and tiltrotor wind-tunnel models [8] pushed toward the introduction of real-time capabilities into the multibody free software MBDyn [9].

The essential feature of this implementation is to show the feasibility of realistic, hardware-in-the-loop simulation capabilities by means of existing, general-purpose free software tools for the real-time scheduling and interprocess communication (Linux RTAI [5]), the control task (RTAILab [2]), and the multibody analysis (MBDyn), with off-the-shelf low-cost hardware.

This peculiar application really stretches general-purpose multibody analysis software to its limits because of the stringent requirements on worst-case solution time for a single time step. As a consequence, the selected software has been carefully tuned in the solution phase, significantly impacting Jacobian assembly and sparse matrix handling to obtain as much performance improvements as possible [11], with very positive impact on non real-time simulation as well.

A critical issue is represented by the need to model friction in joints, which may lead to convergence problems during state transition. For this purpose, specialized dynamic friction models, based on the LuGre model, have been developed and incorporated in MBDyn.

2 DISTRIBUTED REAL-TIME MULTIBODY SIMULATION

Real-time simulation poses very stringent requirements on participating software. As a consequence, it may not be trivial to provide real-time capabilities to existing software.

A preliminary requirement is related to the capabilities of the operating system (OS), which must be able to provide deterministically bounded worst case scheduling latencies. This is not the case for many, if not all, general-purpose OSes, despite occasional claims of somewhat limited real-time capabilities. One remarkable example is the Linux OS, at least up to version 2.4 of the kernel; in any case, even later versions do not provide very stringent scheduling latencies, of the order of the tens of microseconds, as required by very specialistic applications, like high-speed machinery control.

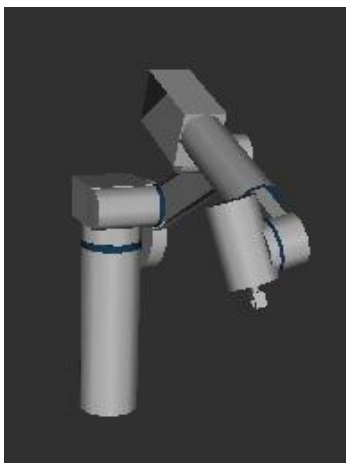

Figure 1: 6 DOF robot

Simulation software may not have such stringent requirements, unless very specialistic applications are addressed. However, hardware-in-the-loop simulation requires the simulation software that emulates the experiment to match the same scheduling requirements of the rest of the system (e.g. A/D, D/A adapters, control systems and so).

Recently, the Real-Time Application Interface (RTAI, `http://www.rtai.org/`) extension to the Linux OS, developed at the "Dipartimento di Ingegneria Aerospaziale" of the University "Politecnico di Milano" and distributed as free software under the GPL license, has gained sufficient maturity to find a clear position in the implementation of industrial scale real-time control systems. It provides a framework consisting in native, as well as POSIX-like system calls within a reliable worst case latency well below a hundred (20-25) microseconds.

Usually, real-time simulation is addressed by means of dedicated software because of its intrinsic needs of speed performances and constant worst-case timestep computational costs. This approach may suffer from lack of generality, because dedicated formulations may not allow a wide variety of model libraries and require specific implementation of features typically available in general purpose software.

In the present work the opposite approach has been attempted, consisting in real-time enabling a general-purpose software. The multibody analysis software MBDyn [9, 10], also developed at the "Dipartimento di Ingegneria Aerospaziale" of the University "Politecnico di Milano", has been used for the essential reason that to the Authors' knowledge it is the only general-purpose multibody analysis software available as free software.

MBDyn is implemented in C++, and exploits a considerable number of standard mathematical libraries. Since RTAI delegates some kernel-related operations to the Linux kernel upon request by the task under execution, to allow MBDyn to be run without incurring in non real-time scheduling all direct system calls had to be disabled or, if essential, they had to be wrapped by the corresponding RTAI interfaces.

In detail, all results output on disk have been disabled, and memory allocation had to be completed before entering the real-time scheduling loop, or, whenever convenient, moved to pre-allocated memory pools.

Finally, essential I/O with other real-time tasks, usually dealt with by regular system sockets in non real-time simulations, has been delegated to RTAI native mailboxes. Despite the naming analogy with UNIX native mailbox infrastructure, the RTAI mailboxes implicitly take care of remote process communication, much like UNIX sockets do. As a consequence, an interprocess communication enabled task can transparently interact with local as well as remote processes, and thus participate in distributed real-time simulations, as illustrated in Figure 2. Local scheduling and communication is entirely handled by RTAI; remote real-time communication occurs via ethernet (most 100Mbit interfaces suffice) and requires an additional layer provided by RTNet (`http://www.rts.uni-hannover.de/rtnet/`).

The remaining changes related to real-time enabling consisted in adding few selected calls to enter hard real-time mode and to synchronize with the rest of the tasks participating in the simulation, either by direct kernel scheduling or by means of semaphores triggered by other processes.

3 REAL-TIME CONTROL

Although all the simulation could be performed inside the multibody analysis software, the real-time simulation system has been essentially designed to allow testing control tasks that can be used to control the actual robot. Thask separation may in any case be advisable, since the real-time execution of the simulation sets very stringent requirements on the execution time of

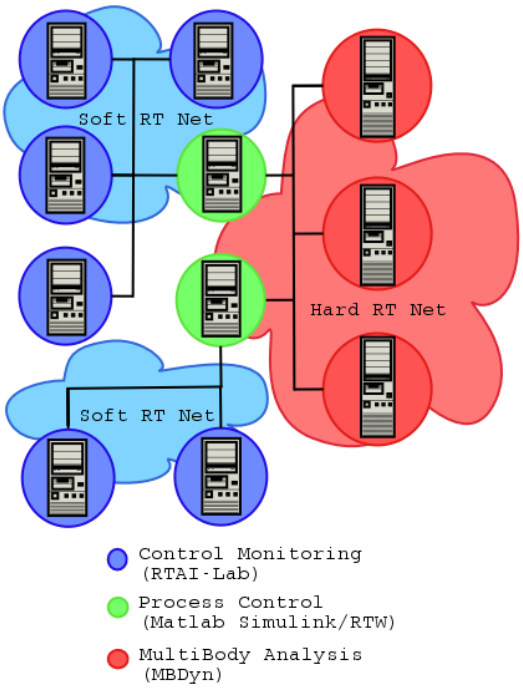

Figure 2: Distributed real-time network layout

each time step during the time integration of the equations of motion of the system.

MBDyn follows an approach based on using redundant coordinate sets, thus integrating the differential equations of motion of each body as if they were unconstrained, plus the algebraic equations that describe the kinematic constraints; the constraint reactions, are applied to the equations of motion by means of Lagrange multipliers.

This approach is quite general and allows to introduce arbitrary topology and arbitrarily formulated configuration dependent forces and kinematic constraints. However, The resulting problem, for a 6 DOF robot, may grow to a hundred equations or more. The integration of differential-algebraic equations (DAE) requires implicit numerical schemes, which in principle have unbounded per-step worst-case numerical cost.

As a consequence, the use of general-purpose software really broadens the possibilities of simulation, but poses stringent requirements and some uncertainties on the feasibility of the approach.

The control of the system is delegated to specialized tasks that result from automatic code generation by commonly used modeling software like Simulink and Scicos. The complete framework allows the fast prototyping of control systems for robotics applications directly from graphical modeling environments without even writing a line of code and without having the robot available yet.

The framework for the real-time execution, monitoring and run-time user interaction of the control tasks is RTAILab [2]; and a screenshot of the interface used for the 6 DOF robot discussed in the applications section is illustrated in Figure 3. RTAILab allows, for instance, to change run-time the parameters of the controller, and to act on the signals that are sent to the simulation, e.g. commands, inputs, disturbances and so, monitoring the selected outputs. The real-time control is performed in hard real-time, i.e. with tight scheduling; the monitoring is performed in soft real-time, because user interaction has less stringent timing requirements.

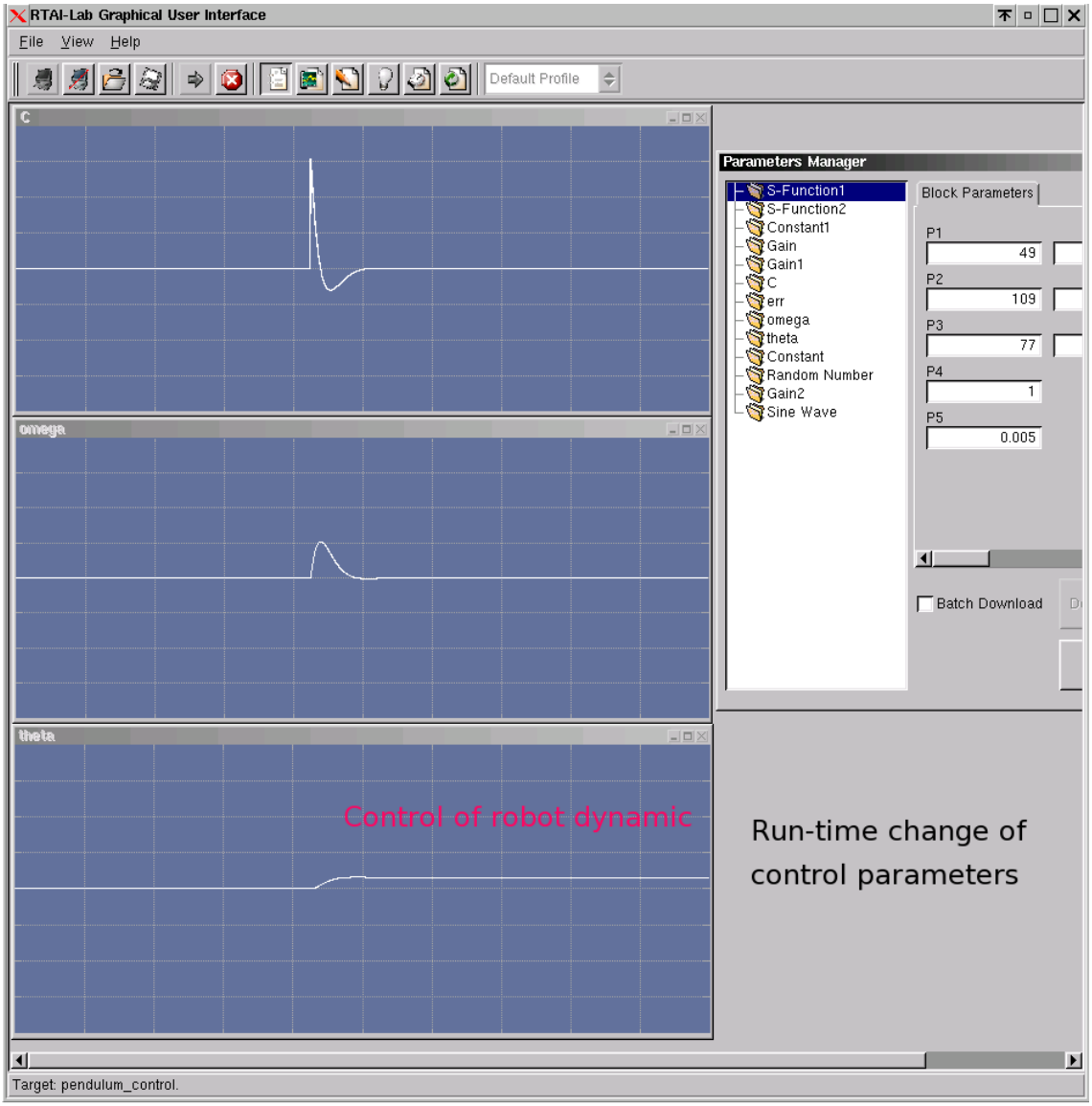

Figure 3: RTAILab graphical user interface

4 COMPUTATIONAL ASPECTS

One major self-imposed constraint in providing multibody software real-time simulation capabilities was that only minor impact or limitation to the batch simulation capabilities was allowed, and by no means features made available in the real-time environment could not be exploited when running in batch mode.

As a result, all performance and interprocess communication improvements that resulted from the satisfaction of the real-time requirements are now beneficial to regular non real-time simulations.

Those that are mostly relevant to the present work are briefly described in the following; they address the performances of the linear solver for "small" and "very small" sparse matrices (between 50 and 2000 equations), and the implementation of realistic joint friction models.

With respect to linear solver performances, MBDyn was designed from the beginning to use arbitrary matrix storage and linear solution methods, because targeted to mechanical and multidisciplinary problems involving elastic systems, thus resulting in matrices that are relatively large for a dense solver and, at the same time, sparse enough to justify the use of stock sparse solvers. However, sparse solvers of general use are typically focused on saving memory occupation at least as much as they are on speeding up computations, because they are targeted for very large problems (10^5–10^6 equations and above).

The default solver used in MBDyn is Umfpack [4] whenever available, or Y12M [13]; the former gives very good performances for problems with few thousands or even tens of thousands equations (typical deformable rotorcraft models analyzed with MBDyn are within 600 and 2000 equations).

There appeared to be room for a dedicated sparse solver that is essentially optimized for speed, at the price of extensive memory usage that may not be critical for problems below 2000 equations [12]. Note that, in this case, extensive memory consumption is not an issue in general terms, but may still impact performances because of cache misses, so the expected advantages in using this solver should be referred to the CPU cache size rather than to the overall RAM size; this consideration can only be accounted for in a heuristic manner, since the way the CPU cache is exploited is not under control of the user.

Table 1 illustrates linear solution times for standard sparse matrix benchmarks [3, 1], where those of Umfpack where column pivoting is computed and used are taken as reference. The name and the size of each problem is reported in the first two columns; column "naïve" contains the results of the linear solver [12] without column pivoting, while columns "UMF p" and "naïve p" contain the timings of Umfpack and [12] when column pivoting is reused from a previous factorization. Column "LA" contain the results obtained with the LAPACK dense solver, as a further reference that may be significant for this class of small matrices. Finally, columns "SLU" and "SLU p" illustrate the results obtained with the SuperLU package [7].

The results appear quite promising in relation to problems of the order of $100 \div 150$ equations, with few notable exceptions for specific sparsity patterns. Table 2 illustrates the overall timing of the multibody analysis with the different solvers when applied to a rigid and a deformable helicopter rotor model and to a rigid 6 DOF robot model.

Friction can severely impact the ability of a controller to precisely position the robot arm. For this reason, the multibody model used to tune the controller accounts for friction and for its most notable effect, stiction. The friction model should be able to reproduce the dependency of the friction coefficient from the relative sliding velocity, e.g. the Stribeck effect. From a computation point of view, real-time simulations require a friction model that does not need

Table 1: Normalized solution time.

	size	naïve	UMF	SLU	LA	naïve p	UMF p	SLU p
bp__1000	822	**0.54**	1	1.47	43.08	**0.21**	0.7	1.12
bp___200	822	**0.6**	1	1.83	86.65	**0.2**	0.66	1.27
bp___600	822	**0.57**	1	1.73	60.44	**0.23**	0.67	1.3
bp_____0	822	**0.99**	**1**	2.62	214.23	**0.36**	0.42	1.55
gre_1107	1107	6.55	**1**	1.45	20.97	2.59	**0.84**	1.32
gre_216a	216	**0.25**	1	0.64	2.17	**0.2**	0.75	0.45
gre_216b	216	**0.26**	1	0.61	2.21	**0.22**	0.74	0.42
gre_343	343	**0.29**	1	0.82	4.59	0.38	0.76	0.62
gre_512	512	**0.34**	1	1	9.03	0.79	0.78	0.84
gre__115	115	**0.24**	1	0.58	0.79	**0.08**	0.68	0.39
gre__185	185	**0.41**	1	0.58	1.4	**0.21**	0.72	0.4
jpwh_991	991	2.82	**1**	2.15	21.23	3.91	**0.82**	1.96
lns__131	131	**0.33**	1	0.68	1.58	**0.09**	0.7	0.42
lns__511	511	4.72	1	**0.92**	10.76	**0.61**	0.77	0.71
lnsp_131	131	**0.27**	1	0.67	1.57	**0.09**	0.7	0.42
lnsp_511	511	0.95	1	**0.92**	10.74	**0.51**	0.77	0.71
mcca	180	**0.24**	1	0.52	1.14	**0.13**	0.63	0.32
mcfe	765	**0.71**	1	**0.79**	5.89	1.63	0.74	**0.42**
nnc261	261	**0.38**	1	0.62	3.11	**0.17**	0.77	0.45
nnc666	666	**0.8**	1	**0.79**	13.2	**0.35**	0.79	0.66
sherman2	1080	10.57	**1**	2.75	11.1	7.42	**0.88**	2.6
shl_0	663	**0.69**	1	4.09	181.37	**0.34**	0.41	3.36
shl_200	663	**0.77**	1	5.2	179.54	**0.36**	0.42	4.42
shl_400	663	**0.81**	1	4.59	177.49	**0.36**	0.42	3.8
str_0	363	**0.58**	1	2.64	38.15	**0.21**	0.4	1.06
str__200	363	**0.29**	1	1.02	10.77	**0.13**	0.64	0.46
str__400	363	**0.31**	1	0.98	9.74	**0.11**	0.67	0.48
str__600	363	**0.35**	1	0.85	7.88	**0.17**	0.66	0.44
west0067	67	**0.13**	1	0.53	0.41	**0.07**	0.63	0.31
west0156	156	**0.42**	1	1.13	5	**0.08**	0.55	0.69

Table 2: Multibody simulations normalized run-time.

	size	naïve p	UMF p	SLU p	LA
rigid helicopter rotor	100	**0.62**	1	1.05	1.08
deformable helicopter rotor	791	**0.9**	1	1	12.46
rigid 6 DOF robot with friction	120	**0.52**	1	0.61	0.9

discrete state transitions, because state transitions often, if not always, require additional Jacobian assembly and matrix refactorization to be accounted for, thus increasing the likelihood of overruns of the multibody code.

A whole class of friction models with internal states has been developed in the past years. All these models deal with stiction without needing discrete state transitions. Among them, the modified LuGre friction model proposed in [6] has been selected. In this particular friction model the friction coefficient is a function of an internal state z, of its derivatives and of the relative velocity $\dot{x}$. A differential equation describes the evolution of the internal state z, leading to the law

$$\begin{aligned} f &= \sigma_0 z + \sigma_1 \dot{z} + \sigma_2 \dot{x}, & \sigma_0, \sigma_1, \sigma_2 > 0 \\ \dot{z} &= \dot{x}\left(1 - \alpha(z,\dot{x})\frac{\sigma_0}{|f_s(\dot{x})|}\frac{\dot{x}}{|\dot{x}|}z\right), \end{aligned} \tag{1}$$

where the function $\alpha(z,\dot{x})$ must satisfy the condition

$$\alpha(z,\dot{x}) = 0 \qquad \forall z \in \{|z| \leq \overline{z}\}, \quad \forall \dot{x} \in \Re. \tag{2}$$

A possible expression for α is

$$\alpha(z,\dot{x}) = \left\{ \begin{array}{rr} 0, & |z| \leq \overline{z} \\ \alpha_m(z,\overline{z},\hat{z}), & |\overline{z}| < |z| < |\hat{z}| \\ 1, & |z| \geq |\hat{z}| \end{array} \right\} \begin{array}{l} \text{sgn}\,(\dot{x}) = \text{sgn}\,(z) \\ \\ \\ \text{sgn}\,(\dot{x}) \neq \text{sgn}\,(z) \end{array} \tag{3}$$

where $\hat{z} = |f_s(\dot{x})|/\sigma_0$ is the elastic relative displacement at stiction, and the function $\alpha_m(z,\overline{z},\hat{z})$ is shown in Figure 4. The resulting friction model can reproduce stiction, elastic relative dis-

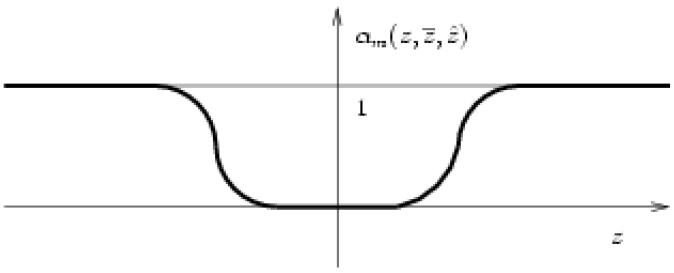

Figure 4: $\alpha(z,\dot{x})$ for $\text{sgn}\,(\dot{x}) = \text{sgn}\,(z)$

placements before sliding, the Stribeck effect and the memory effect that has been observed in sliding contacts with velocity variations. This model can be linearized analytically, thus allowing for a fast and robust implicit time integration of the equations of motion even in presence of friction.

5 APPLICATION: 6 DOF ROBOT

The real-time simulation framework that is discussed in this paper has been applied to the simulation of an industrial 6 DOF robot. The model is built with seven rigid bodies: six for the robot parts, and the last one to model the robot ground support. Six revolute hinges connect

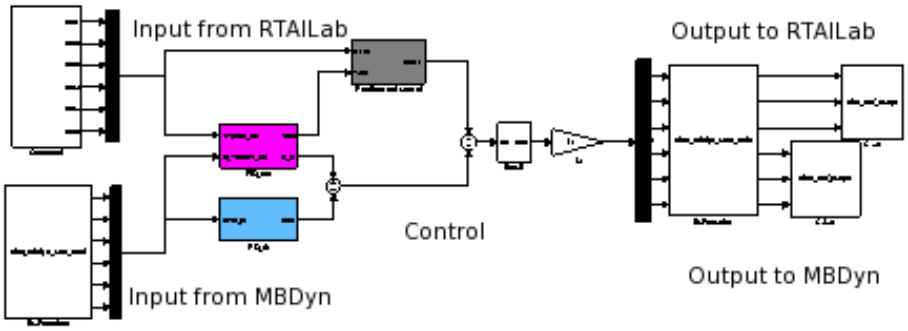

Figure 5: Control layout

the robot arms to each other and to the support. The ground support is constrained by a clamp joint. The model constrained dynamics is described by 114 equations, which grow to 120 when the dynamic friction model is introduced in each hinge. An internal couple about each hinge axis is applied to mimic the torque provided by the motors. The values of the relative rotation and angular velocity between connecting bodies is sent to an outbound real-time mailbox. An inbound real-time mailbox is used to read the values of the internal couple that must be applied by the motors. On an AMD Athlon XP 2400+, whose internal clock is about 2 GHz, the multibody code can simulate the dynamics of the robot model without friction with a frequency up to 2.4 kHz; with friction this figure drops to 1.7 kHz; the drop is dictated by the need for extra iterations to converge at each time step rather than by the slight increase in model size. These figures allow to connect the multibody code to a controller with a frequency of 1 kHz, while preserving enough spare time to reduce the likelihood of overruns of the multibody code when convergence becomes critical. Note that few overruns in general are acceptable, as they may be essentially considered as disturbances from the point of view of the controller, since the output of the simulation should be regular enough to allow, in case of overrun, to reuse the measures at the previous time step.

The control program can be built using Simulink or Scicos, and is run as a separate task, possibly on a dedicated PC. It receives the relative rotations and angular velocities from MBDyn via real-time mailboxes; it may also receive selected control parameters, like gains, desired inputs and so, from an RTAILab process. The control internal couples are computed, and fed back to MBDyn. The control program can also send data to RTAILab, allowing the remote monitoring of the robot state, as shown in Figure 5.

The typical layout of the real-time simulation system is show in Figure 6. A dedicated PC is used for the multibody simulation of the robot, the most computationally intensive process. Another configuration that has been used in a related research is made of dual-CPU PCs, where process confinement is used to force one CPU to run the simulation, while the other is used to run the control and, since this task is supposed to be less resource consuming, the monitoring and user-interaction interface.

6 CONCLUSIONS

The real-time variant of the general-purpose multibody analysis software MBDyn, based on the services provided by the RTAI application interface to the Linux OS and integrated into a distributed real-time simulation environment, represents a valid instrument for the development and testing of systems because on the one hand it allows, in principle, the modeling freedom of general-purpose simulation software; on the other hand it allows to test sophisticated con-

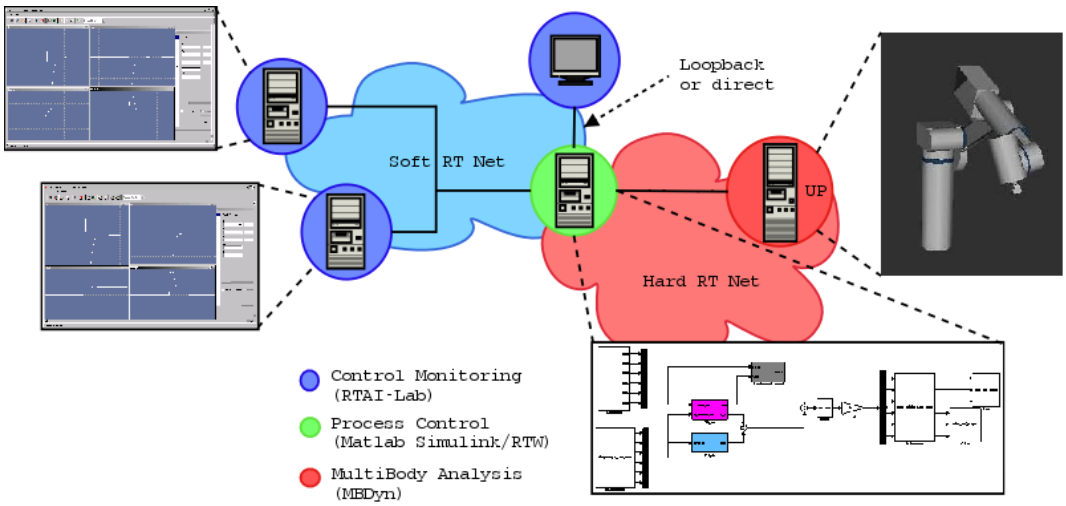

Figure 6: Distributed real-time simulation layout

trol systems without requiring the presence of the real hardware, thus potentially saving time, resources, and avoiding critical operating conditions.

The present work illustrates how the recent development of the software increased the modeling capabilities, by allowing higher sampling rates and larger models. The introduction of realistic friction models allows to account for details of robot dynamics that are critical in the design of control systems; the experimental verification of the capability to model this type of forces without incurring in overruns illustrates how the theoretical disadvantages of using implicit integration schemes and a general-purpose simulation approach may not represent a design limit.

ACKNOWLEDGEMENT

The Authors acknowledge the contribution of Michele Attolico and Matteo Martegani to the development of the project.

REFERENCES

[1] R. Boisvert. Matrix market: A web resource for test matrix collections. In *Townmeeting on Online Delivery of NIST Reference Data*, 1997. http://math.nist.gov/MatrixMarket/.

[2] R. Bucher and L. Dozio. CACSD under RTAI linux with RTAI-LAB. In *Fifth Real-Time Linux Workshop*, Valencia, Spain, November 9–11 2003.

[3] T. A. Davis. University of florida sparse matrix collection. *NA Digest*, 92(42), October 16 1994. `http://www.cise.ufl.edu/research/sparse/matrices/`.

[4] T. A. Davis. Algorithm 832: Umfpack, an unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software*, 30(2):196–199, June 2004.

[5] L. Dozio and P. Mantegazza. Real time distributed control systems using RTAI. In *Sixth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC)*, Hakodate, Hokkaido, Japan, May 14–16 2003.

[6] P. Dupont, V. Hayward, B. Armstrong, and F. Altpeter. Single state elastoplastic friction models. *IEEE Transactions on Automatic Control*, 47(5):787–792, May 2002.

[7] X. S. Li and J. W. Demmel. SuperLU-DIST: A scalable distributed-memory sparse direct solver for unsymmetric linear systems. *ACM Transactions on Mathematical Software*, 29(2):110–140, June 2003. http://doi.acm.org/10.1145/779359.779361.

[8] P. Masarati, M. Attolico, M. W. Nixon, and P. Mantegazza. Real-time multibody analysis of wind-tunnel rotorcraft models for virtual experiment purposes. In *AHS* 4^{th} *Decennial Specialists' Conference on Aeromechanics*, Fisherman's Wharf, San Francisco, CA, January 21–23 2004.

[9] P. Masarati, M. Morandini, G. Quaranta, and P. Mantegazza. Open-source multibody analysis software. In *Multibody Dynamics 2003, International Conference on Advances in Computational Multibody Dynamics*, Lisboa, Portugal, July 1–4 2003.

[10] P. Masarati, M. Morandini, G. Quaranta, and P. Mantegazza. Computational aspects and recent improvements in the open-source multibody analysis software "MBDyn". In *Multibody Dynamics 2005, ECCOMAS Thematic Conference*, Madrid, Spain, June 21–24 2005.

[11] M. Morandini and P. Mantegazza. A naive linear solver for small sparse linear systems. to be published.

[12] M. Morandini and P. Mantegazza. Using dense storage to solve small sparse linear systems. Submitted to *ACM Transactions on Mathematical Software* (ACM TOMS).

[13] Z. Zlatev, J. Wasniewski, and K. Schaumburg. *Y12M - solution of large and sparse systems of linear algebraic equations*. Springer-Verlag, Berlin-Heidelberg-New York, 1981.