POLITECNICO DI MILANO

A real-time hardware-in-the-loop simulator for robotics applications

Marco Morandini, Pierangelo Masarati and Paolo Mantegazza

Outline

1. Objectives
2. Approach
3. Computational aspects
4. Application
5. Conclusions

Objectives

Develop a General Purpose Real-Time experiment simulator:

- hard real-time capabilities
- minimal modeling limitations
- commonality of modeling

Develop real-time models and controls for a wide spectrum of applications

- model accuracy
- analysis accuracy
- distributed control Real-time simulation and control
- Robot simulation: 6 dofs, 1 kHz

Approach

Requirements:

- use generic libraries (algebra, communication, ...)
- use general purpose simulation software
- use/ develop Open- Source software
- run on "low" cost platforms (dual Athlon)

Real time: RTAI

Multibody: MBDyn

Communication: RTNet

Acquisition and control: RTAILab

Real- time simulation

Traditional real- time simulations:

- minimal set (ODE)
- explicit integration
- specialized topology
  bound to OS internals

→ fast, fast, fast!
difficult to extend
code/ tool duplication

Proposed real- time solution:

- Redundant set (DAE)
- Implicit integration
- General topology
- POSIX compliant

→ Performance limitations
code/ tool commonality
Modeling flexibility

From conventional to Real- time code

Conventional simulation software

~180,000 LOC

Avoid / wrap system calls (disk writes)

Statically preserve stack / initialize resources

STL containers: memory pools

Insert few task execution / control statements

Add real- time I/O communication provisions (mailboxes)

Real-time MB code

Real- time control and monitoring

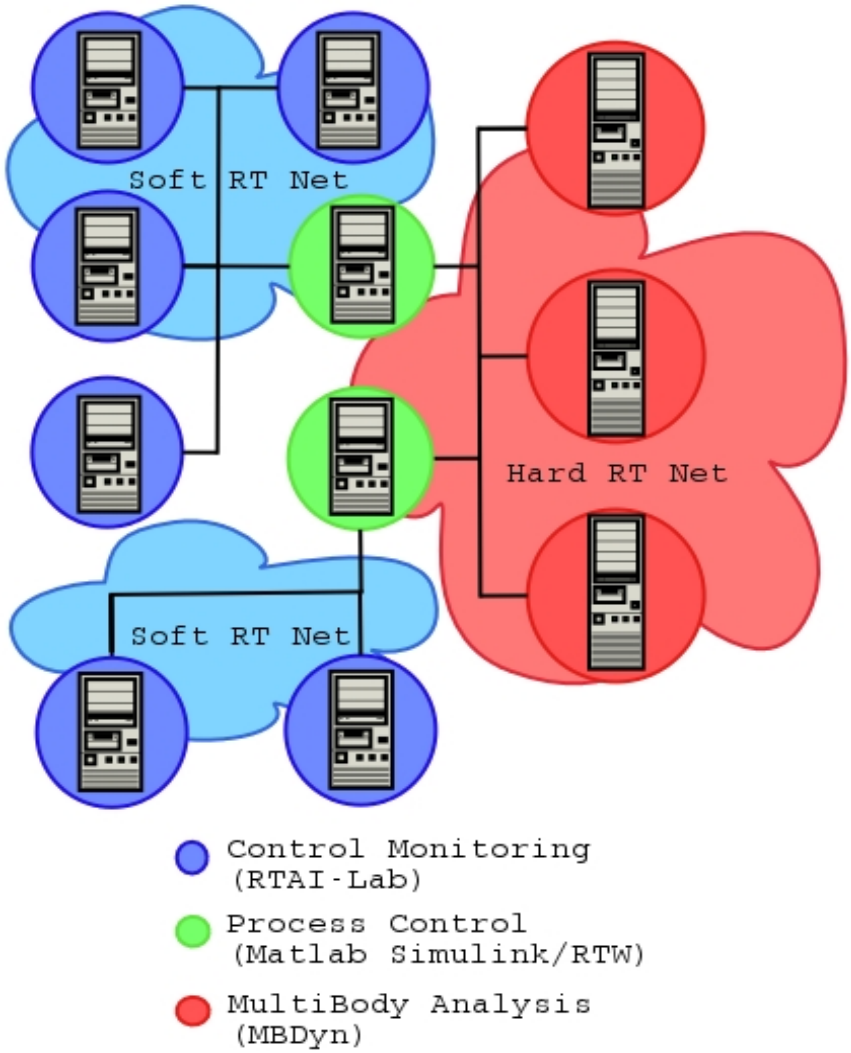
Distributed Real- time software:

- hard Rtnet for control loop
- soft Rtnet for monitoring

Control: simulink/ scicos

Simulation: MBDyn

Monitoring: RTAI- Lab



Computational issues - 1

Mbdyn: redundant set

implicit integration

nonlinear equations

- Modified Newton-Raphson
- “Small”, sparse, asymmetric linear system

New, dedicated sparse solver -> fast!

- linear solver: 2~3 times faster (up to 3000 unknowns)
- simulation code: 2 times faster (100~120 unknowns)

Computational issues - 2

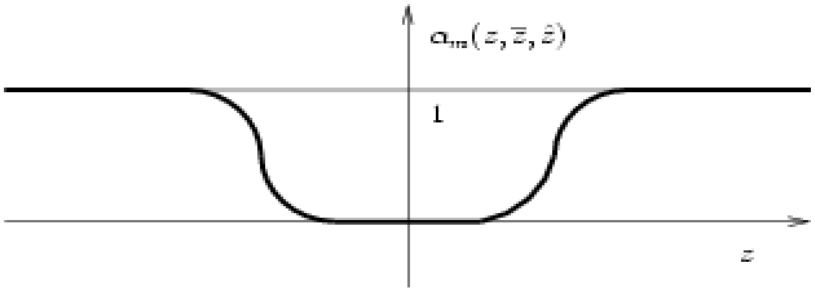
Friction: continuous, differentiable model

- no trial-error during Newton-Raphson
- differentiable
- Stiction, elastic pre-sliding, Stribeck effect, memory

Modified LuGre (DuPont, Hayward, Armstrong and Alpeter)

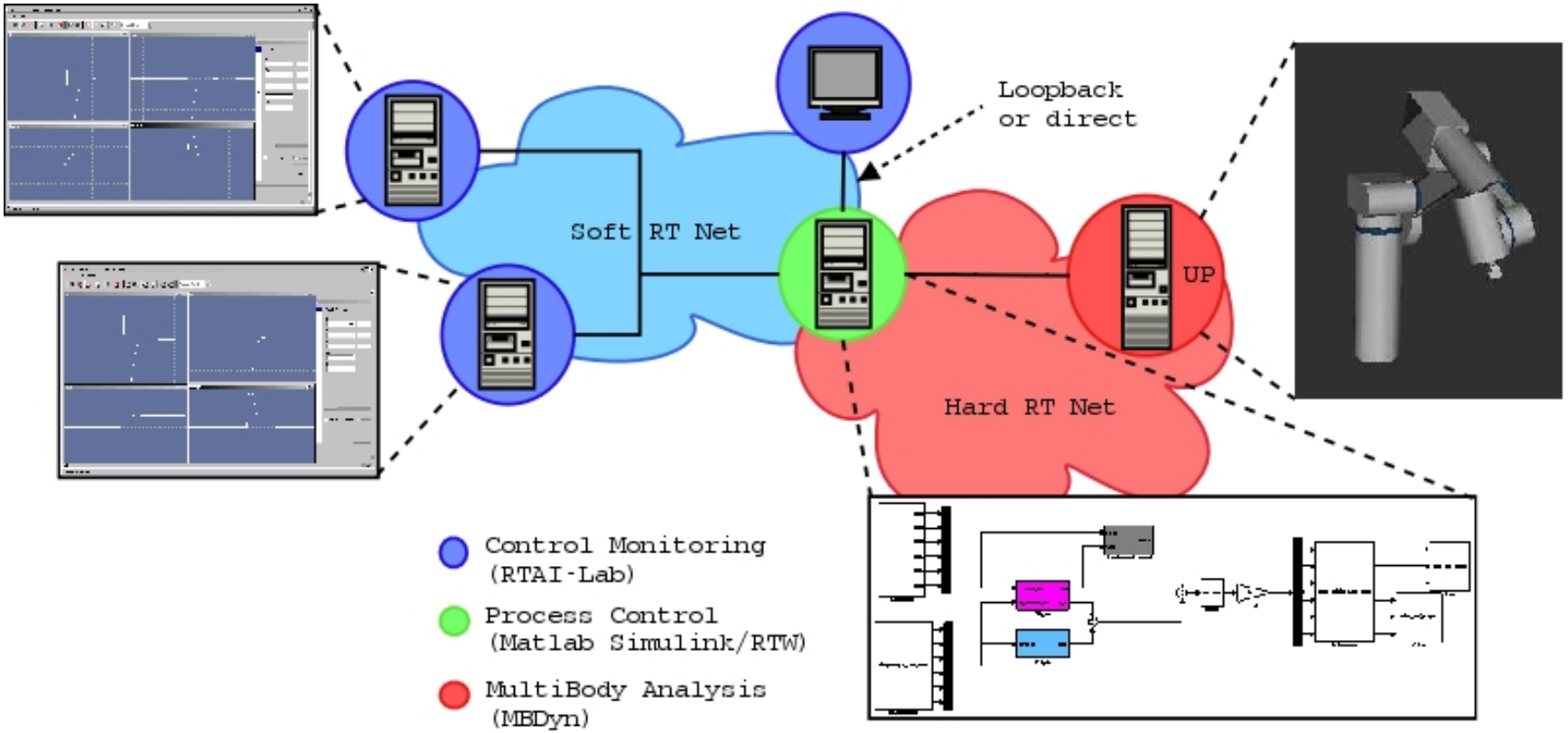
$$f = \sigma_0 z + \sigma_1 \dot{z} + \sigma_2 \dot{x}$$

$$\dot{z} = \dot{x}\left(1 - \alpha(z, \dot{x}) \frac{\sigma_0}{|f_s(\dot{x})|} \frac{\dot{x}}{|\dot{x}|} z\right)$$



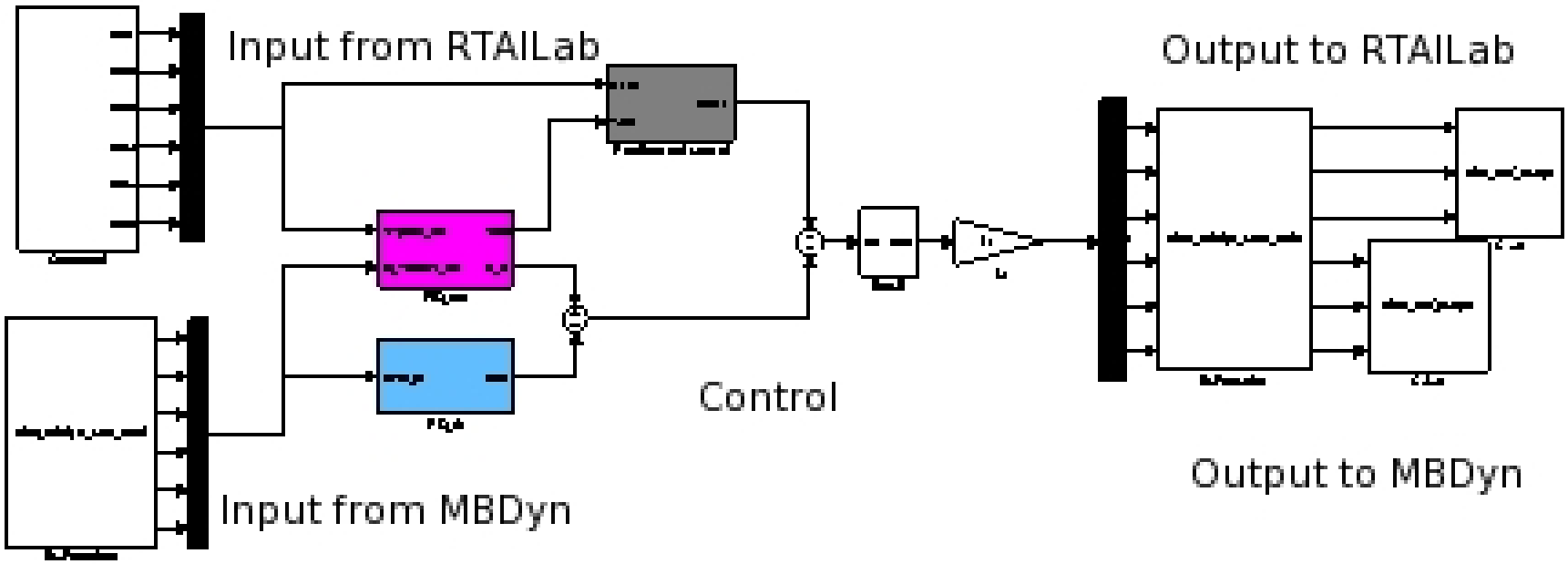
Application: 6 DOF robot

Overall setup



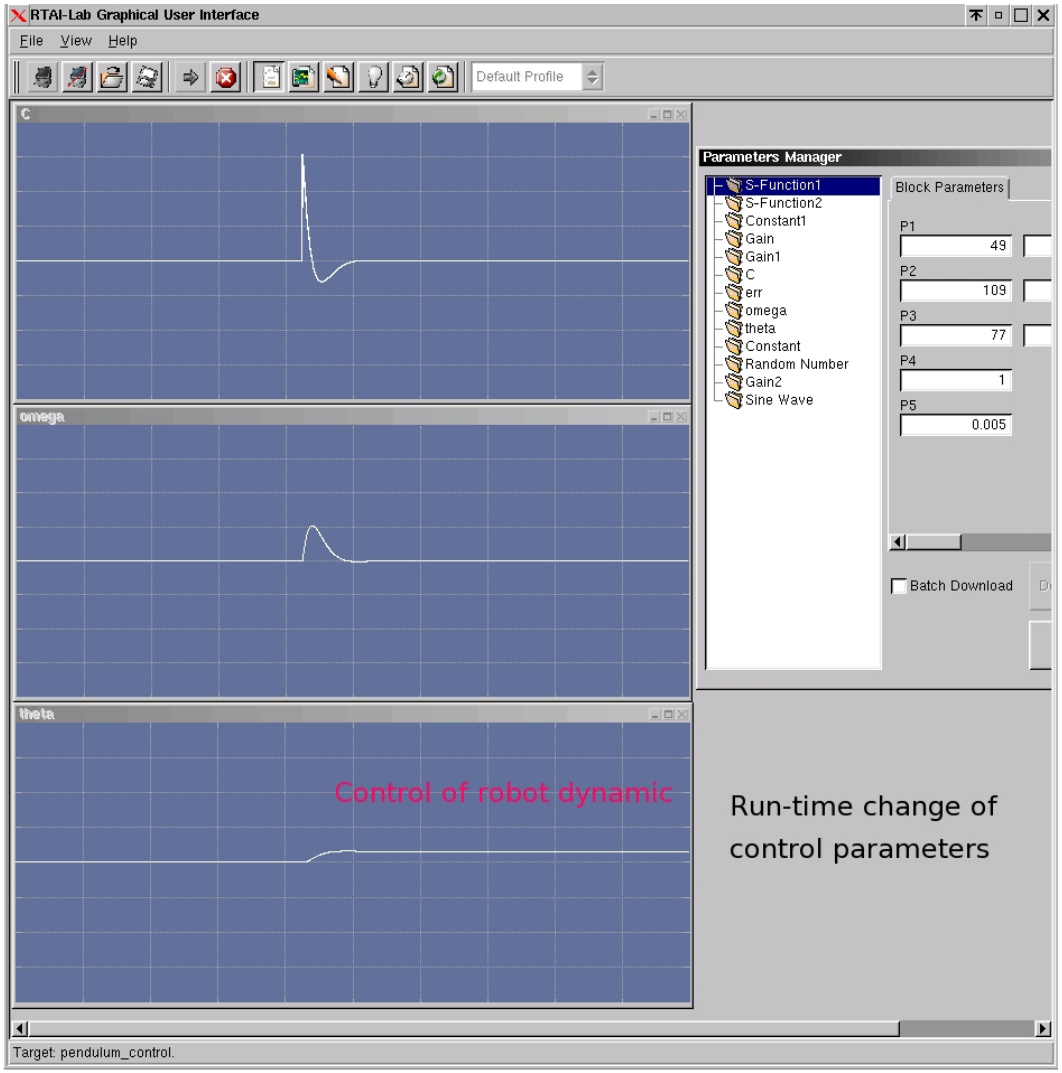
Application: 6 DOF robot

Control communications



Application: 6 DOF robot

RTAI- Lab



Conclusions and future developments

Minimal impact on the code

General interface

Good performances, even with friction

- Athlon 2.4: 2.4 kHz without friction, 1.7 with friction

Same tool for project, simulations, feed-forward, ...

Feed-forward

Hardware in the loop

...