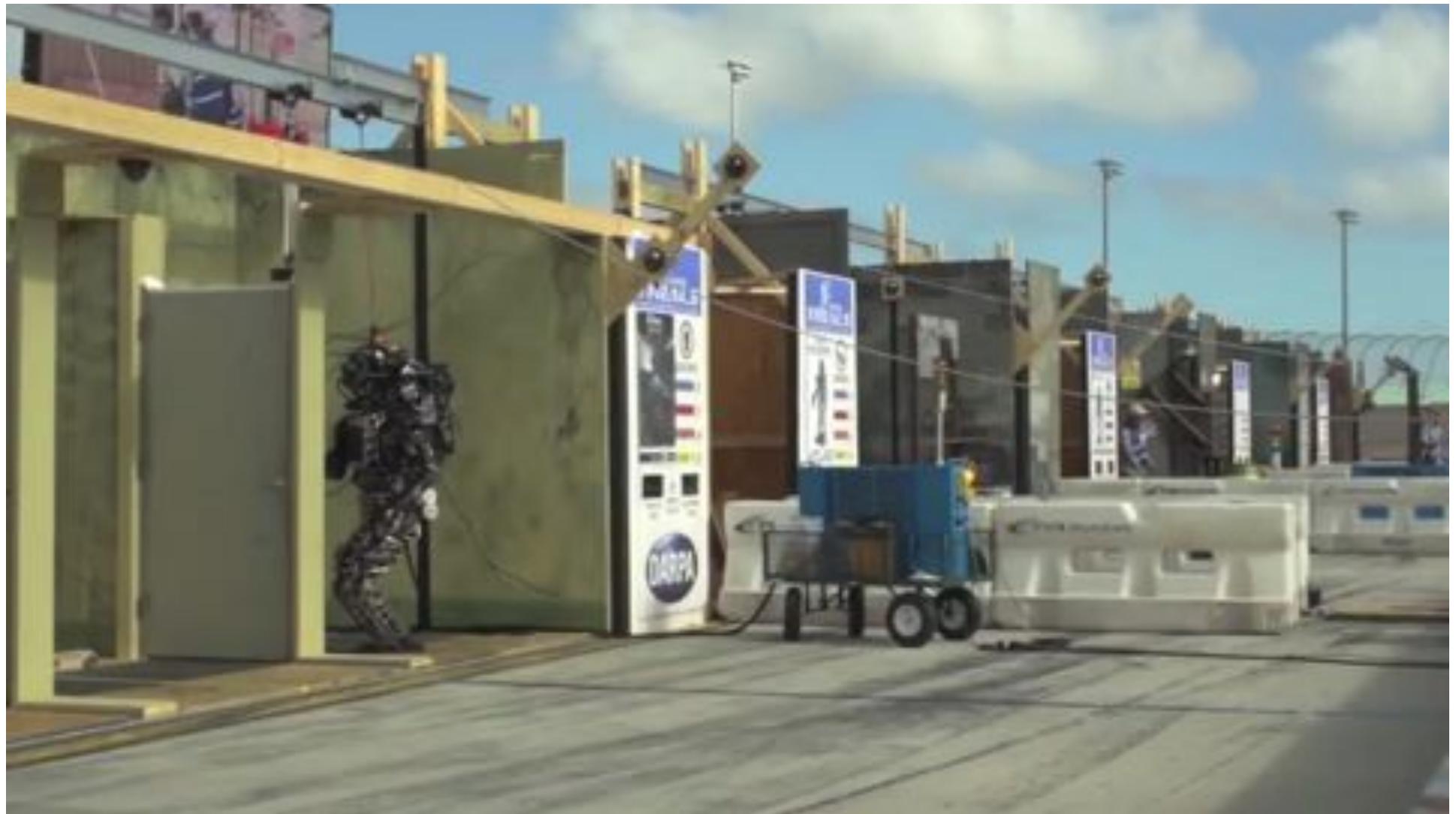


Humanoid whole-body motion control with multiple contacts

Francesco Nori
Robotics, Brain and Cognitive Sciences

Robot-Environment interaction



The goal



Dynamics



Dynamics matter



Control



Contact force sensing



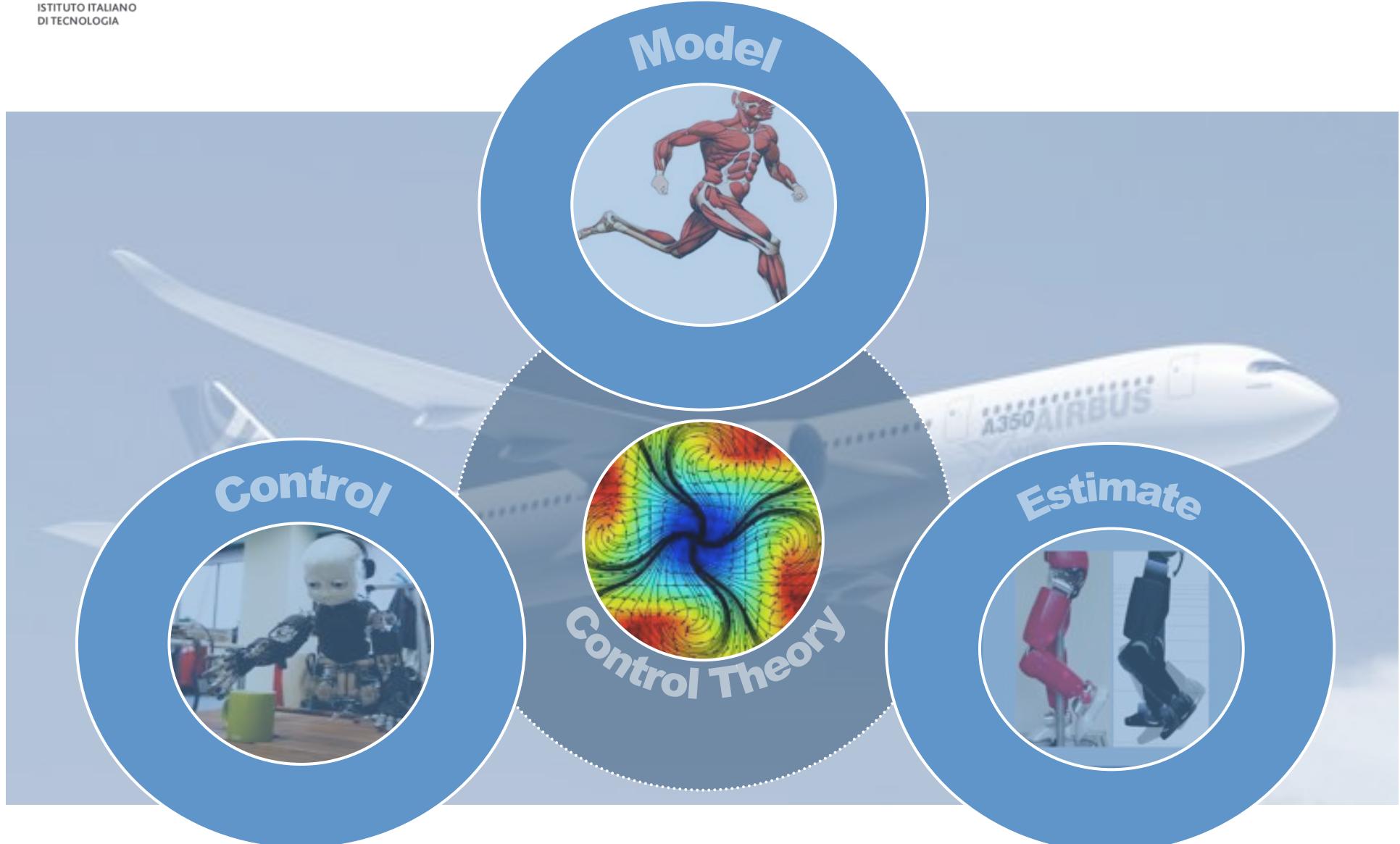
FP7-EU project CoDyCo



FP7-EU project Koroibot



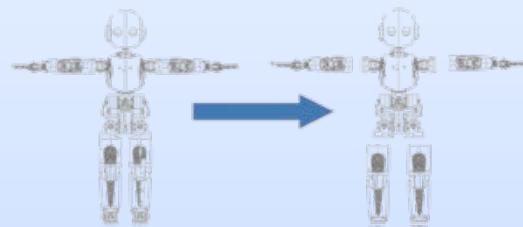
Control theory as a tool for basic science



Basic science



Identification

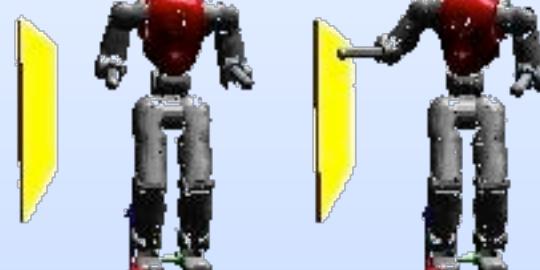


Parametric approach to whole-body dynamics identification

Avoiding to rely on Inertial Parameters in Estimating Joint Torques with proximal F/T sensing. Traversaro et al.



Linearization

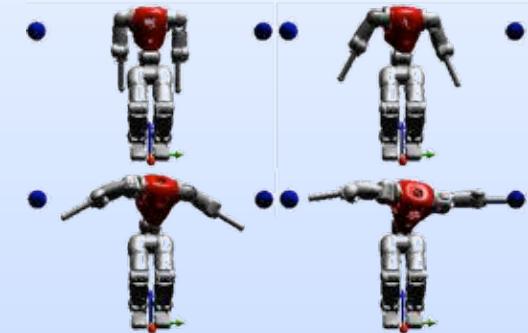


Feedback linearization of constrained mechanical systems.

Partial Force Control of Constrained Floating-Base Robots. Del Prete et al.



Optimal control



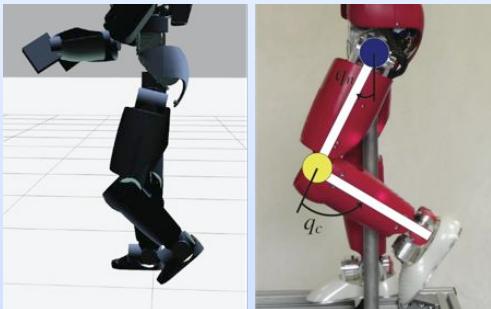
A hierarchical differential dynamic programming approach to optimal control.

Prioritized Optimal Control: a Hierarchical Dynamic Programming approach. Romano et al.

Basic science



Adaptive control

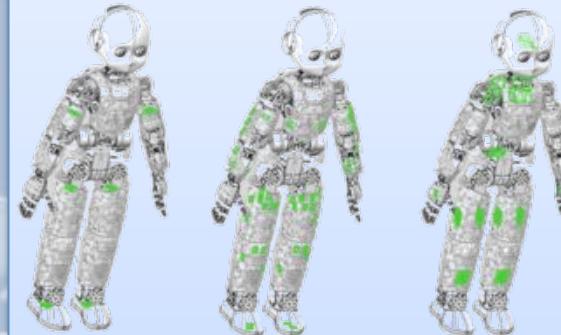


Adaptive control of under actuated mechanical systems.

Collocated Adaptive Control of Underactuated Mechanical Systems. F. Romano, D. Pucci, F. Nori



Estimation

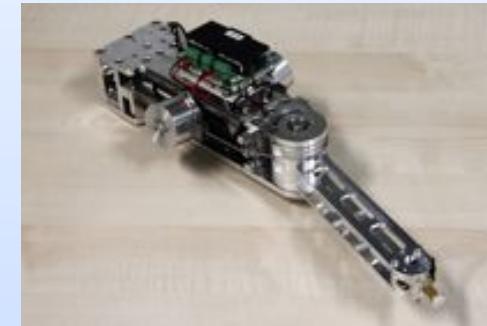


Computing whole-body dynamics with redundant measurements.

BERDY: Bayesian Estimation of Robot Dynamic variables. F. Nori et al.



Compliance



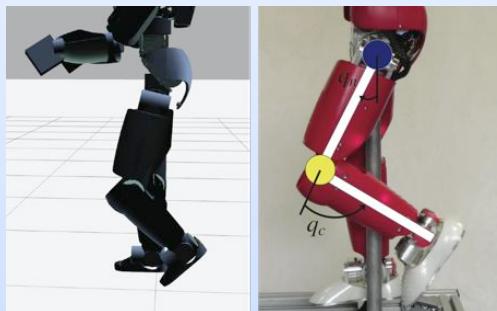
Path integral approaches to the control of compliant actuators.

Stiction Compensation in Variable Stiffness Actuators . Fiorio et al.

Basic science



Torque control

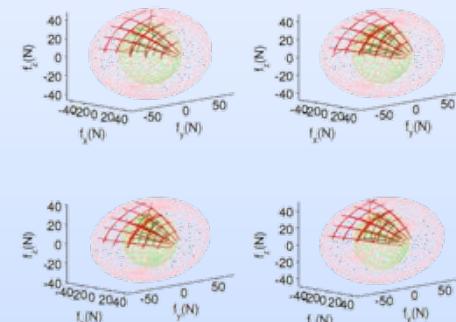


Adaptive control of underactuated mechanical systems.

Collocated Adaptive Control of Underactuated Mechanical Systems. F. Romano, D. Pucci, F. Nori



Sensor calibration

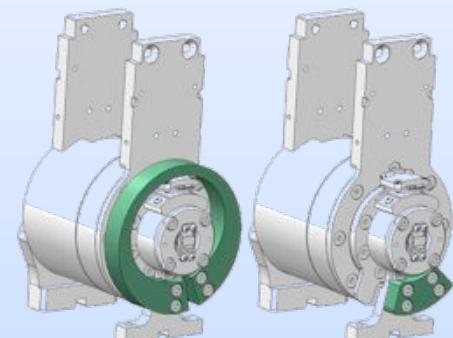


In-situ force-torque sensor calibration.

In Situ Calibration of Six-Axes Force Torque Sensors using Accelerometer Measurements. Traversaro et al.



Compliance



Exploitation of compliance in push recovery.

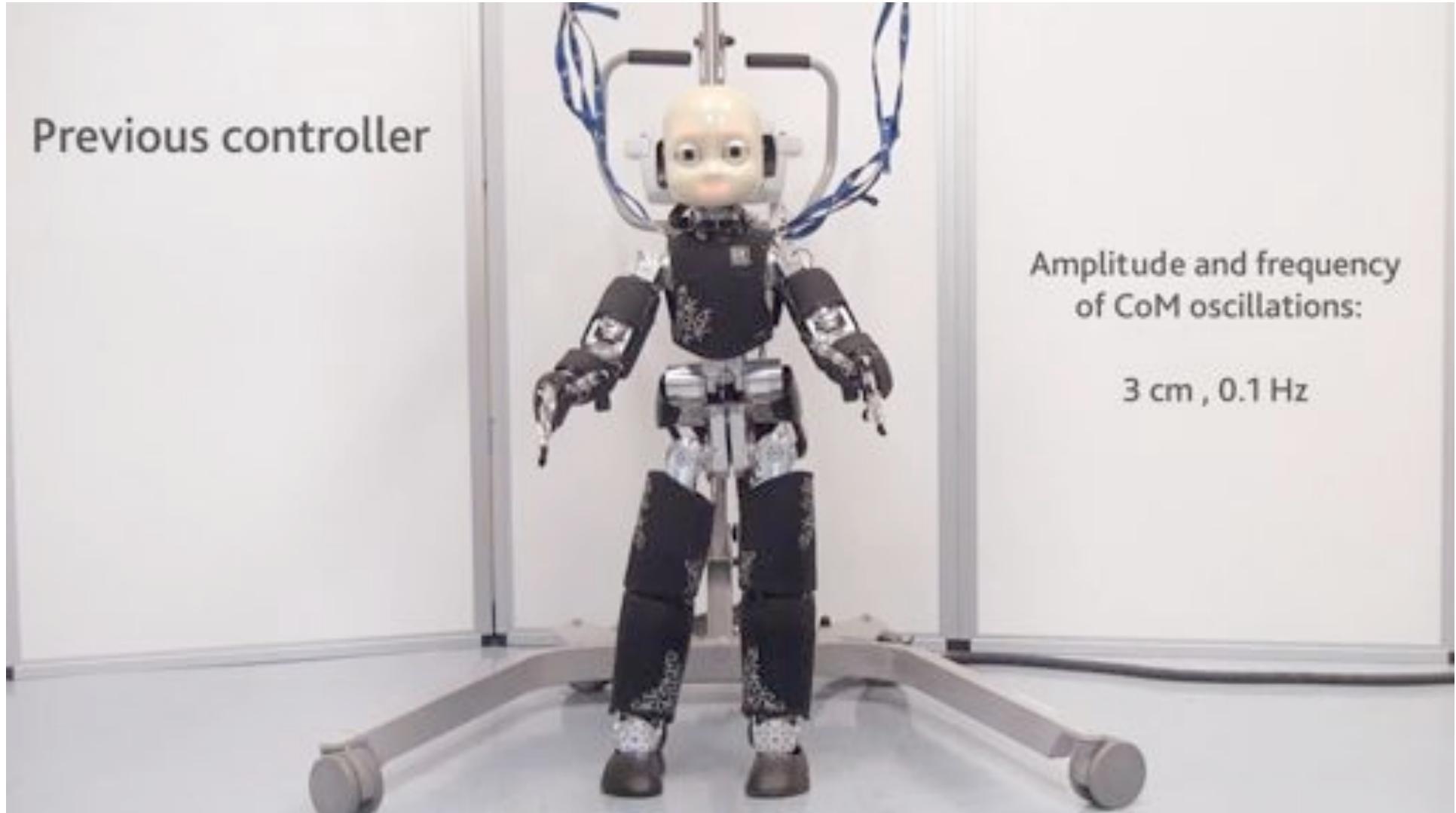
Quantitative Evaluation of Standing Stabilization Using Stiff and Compliant Actuators. Eljiak et al.

CoDyCo results

Previous controller

Amplitude and frequency
of CoM oscillations:

3 cm , 0.1 Hz



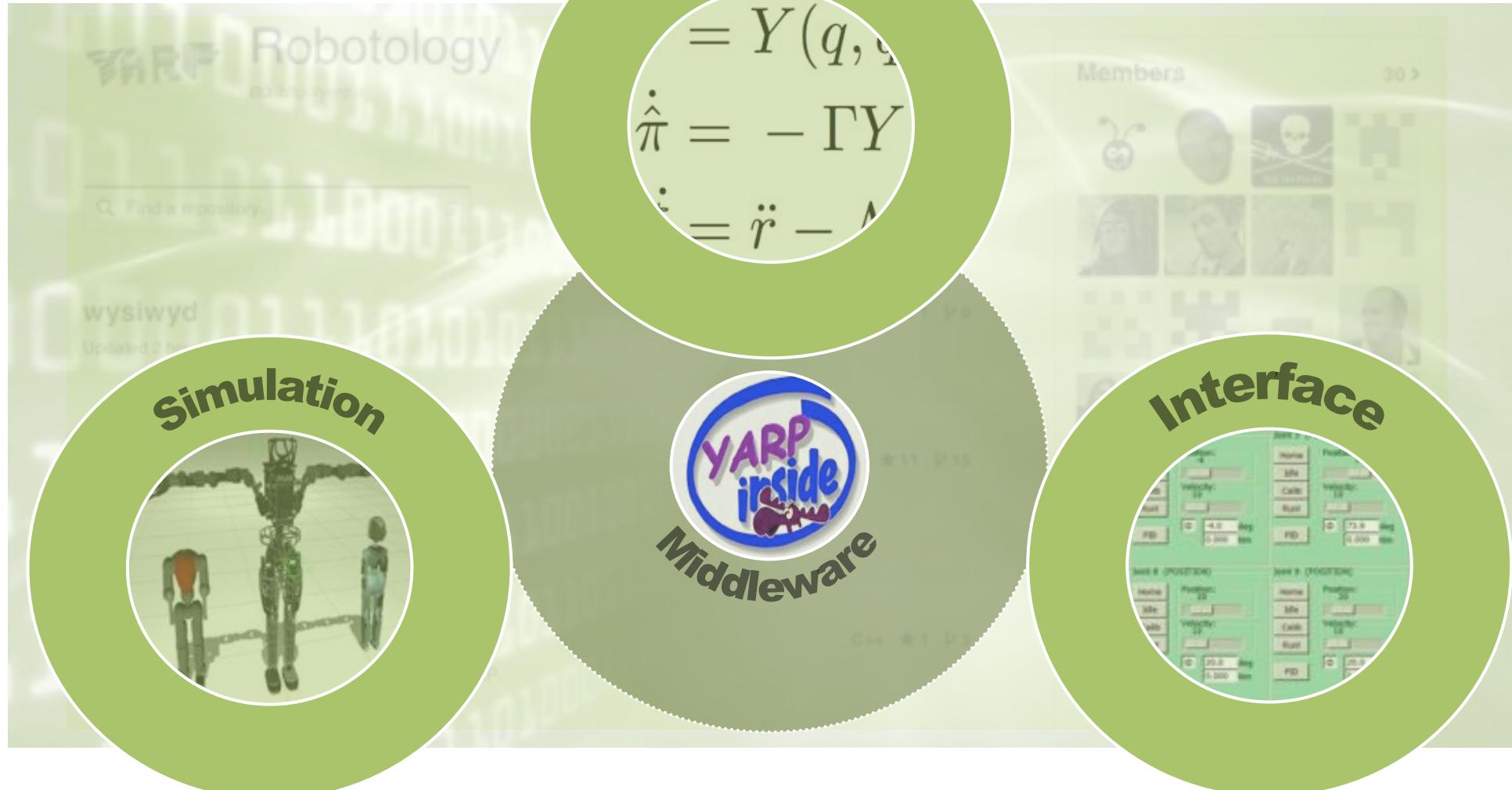
IIT-common (control) software infrastructure for robots

RBCS: S. Traversaro, J. Eljaik, F. Romano, D. Pucci, N. Kuppuswamy

iCub Facility: A. Cardellino, D. Domenichelli, G. Metta, L. Natale

ADVR: A. Rocchi, M. Ferrati, E. Mingo Hoffman, A. Settimi, N. Tsagarakis, A. Bicchi.

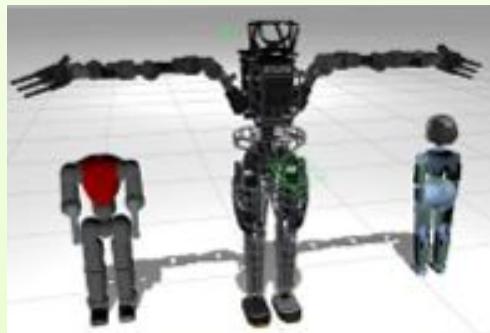
Open source (control) software



Control Software



YARP-Gazebo

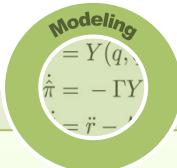


Models of under actuated constrained mechanical systems.

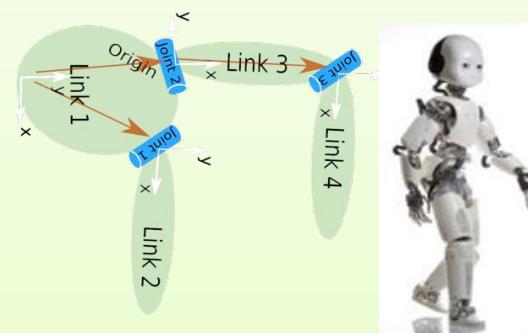
$$M(q)\ddot{q} + h(q, \dot{q}) - J_c^\top f = S^\top \tau$$

$$J_c(q)\ddot{q} + \dot{J}_c\dot{q} = 0$$

A Yarp based plugin for Gazebo Simulator.
Mingo et al.

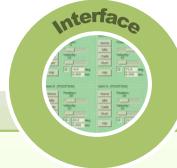


iDynTree

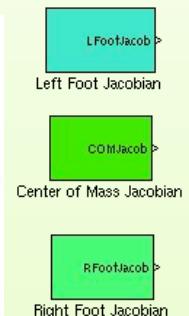
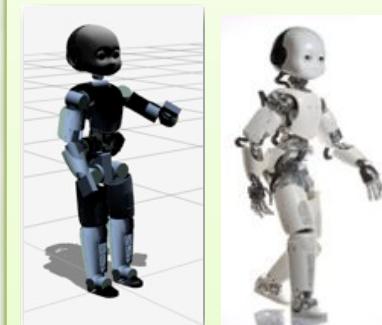


Library for identifying whole-body dynamics in free floating robots.

Inertial Parameter Identification Including Friction and Motor Dynamics. S. Traversaro et al.



wholeBodyInterface



A light abstraction layer for controlling free floating robots.

WBI Toolbox (WBI-T): A Simulink Wrapper for Robot Whole Body Control. J. Eljaik et al.

Outline

- T1.1 Software architecture:
 - The WBI (wholeBodyInterface)
 - The WBIToolbox (simulink interface to the WBI)
- T1.2 Simulator:
 - The YARP interface to Gazebo (gazebo_yarp_plugin)
- T1.4 System dynamics estimation software:
 - The iDynTree::computeRegressor() function
- T1.5 Extension and enhancement of the iDyn library
 - The wholeBodyDynamics software module

Outline

- T1.1 Software architecture:
 - The WBI (wholeBodyInterface)
 - The WBIToolbox (simulink interface to the WBI)
- T1.2 Simulator:
 - The YARP interface to Gazebo
(gazebo_yarp_plugin)
- T1.4 System dynamics estimation software:
 - The iDynTree::computeRegressor() function
- T1.5 Extension and enhancement of the iDyn library
 - The wholeBodyDynamics software module

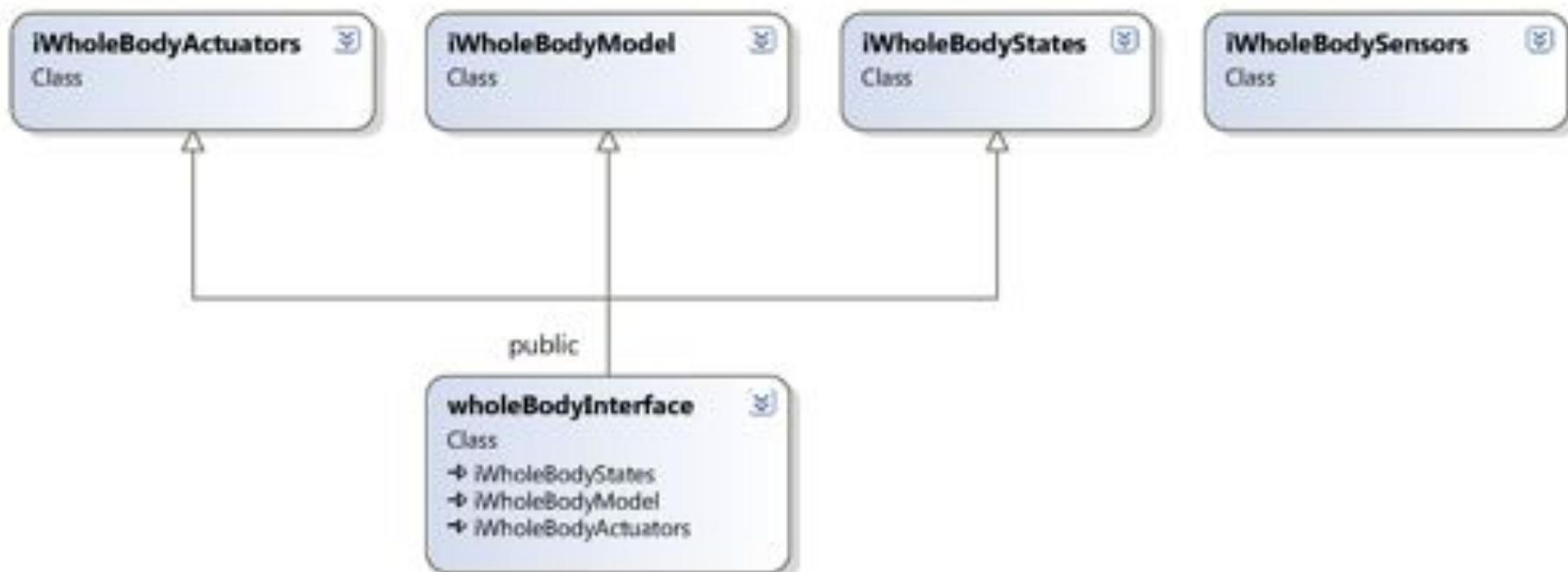
Outline

- T1.1 Software architecture:
 - The WBI (wholeBodyInterface)
 - The WBIToolbox (simulink interface to the WBI)
- T1.2 Simulator:
 - The YARP interface to Gazebo (gazebo_yarp_plugin)
- T1.4 System dynamics estimation software:
 - The iDynTree::computeRegressor() function
- T1.5 Extension and enhancement of the iDyn library
 - The wholeBodyDynamics software module

wholeBodyInterface

- Write once, run on any robot.
- Implemented for the iCub with dependencies: eigen (algebra), KDL (kinematics and dynamics), YARP (interface to the robot).
- Tested on several modules (e.g. wbiToolbox, jointTorqueControl).
- Open-source and available on <https://github.com>.

wholeBodyInterface: structure



Code Example

```
// read state estimations
bool res = robot->getEstimates(ESTIMATE_JOINT_POS,
res = res && robot->getEstimates(ESTIMATE_JOINT_VEL,
res = res && robot->getEstimates(ESTIMATE_BASE_POS,
qRad.data());
dqJ.data();
H_w2b.data());
```

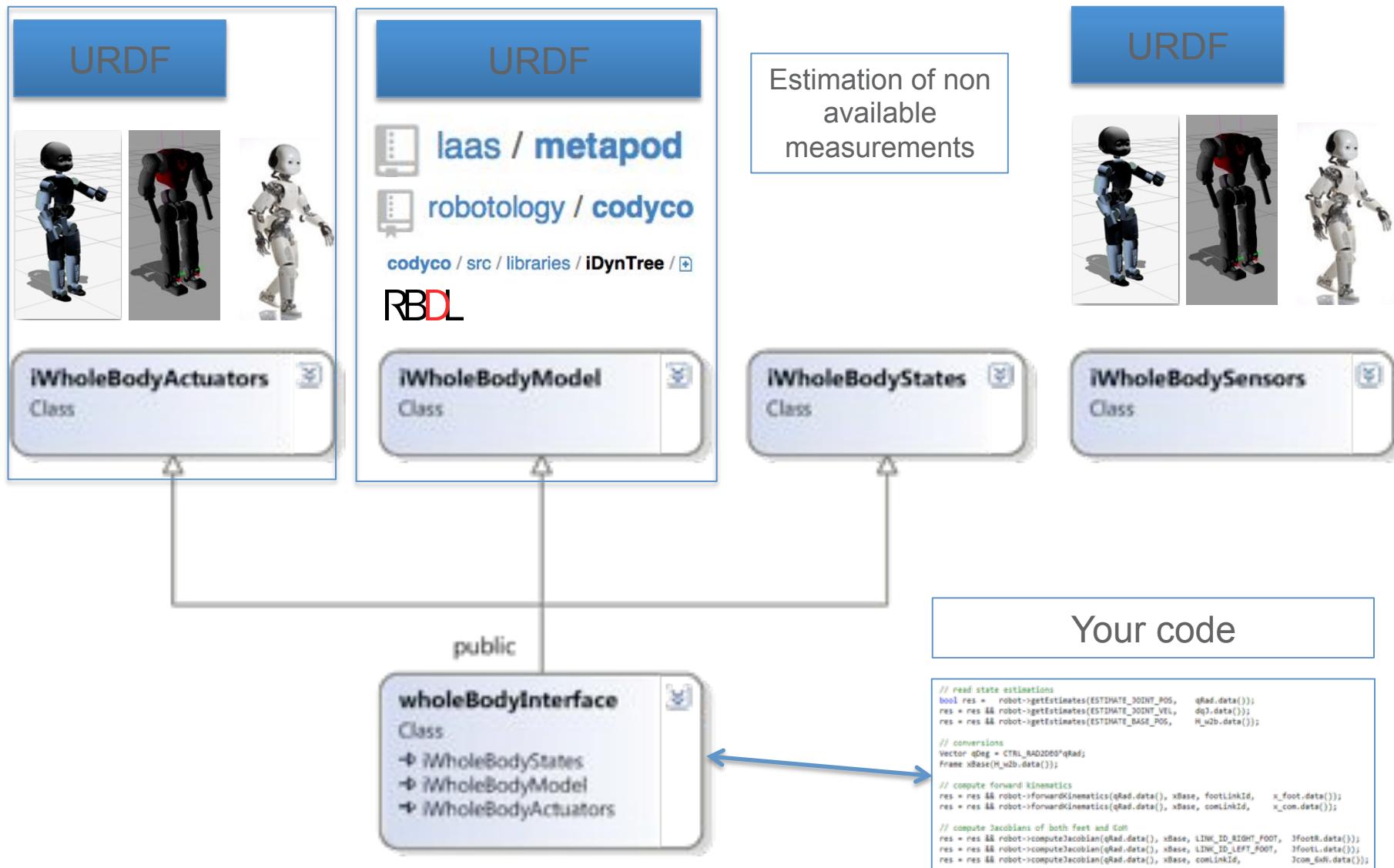
Pointer to double

```
// conversions
Vector qDeg = CTRL_RAD2DEG*qRad;
Frame xBase(H_w2b.data());
```

```
// compute forward kinematics
res = res && robot->forwardKinematics(qRad.data(), xBase, footLinkId, x_foot.data());
res = res && robot->forwardKinematics(qRad.data(), xBase, comLinkId, x_com.data());
```

```
// compute Jacobians of both feet and CoM
res = res && robot->computeJacobian(qRad.data(), xBase, LINK_ID_RIGHT FOOT, JfootR.data());
res = res && robot->computeJacobian(qRad.data(), xBase, LINK_ID_LEFT FOOT, JfootL.data());
res = res && robot->computeJacobian(qRad.data(), xBase, comLinkId, Jcom_GoM.data());
```

Whole-Body Interface



Outline

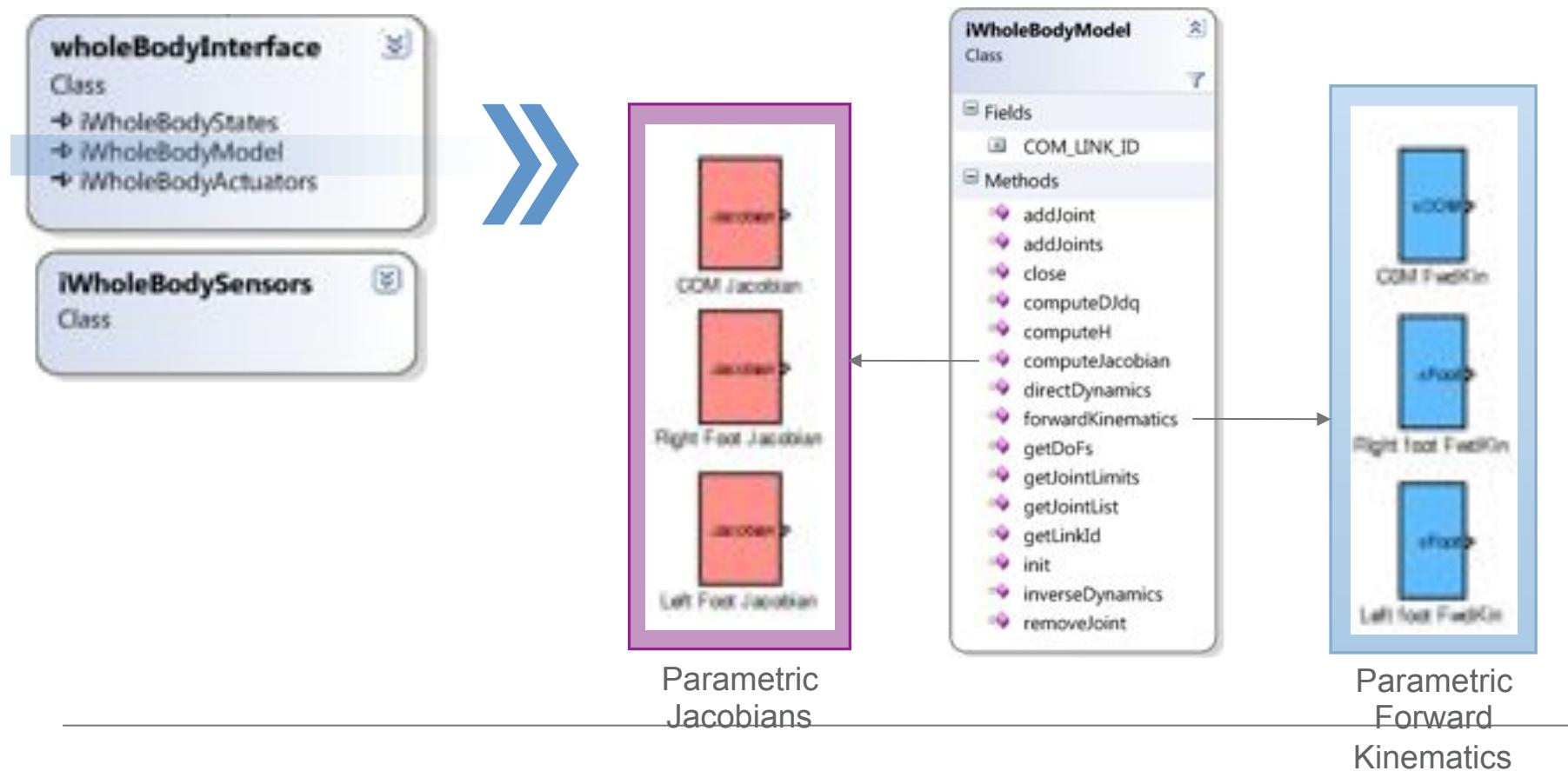
- T1.1 Software architecture:
 - The WBI (wholeBodyInterface)
 - The WBIToolbox (simulink interface to the WBI)
- T1.2 Simulator:
 - The YARP interface to Gazebo (gazebo_yarp_plugin)
- T1.4 System dynamics estimation software:
 - The iDynTree::computeRegressor() function
- T1.5 Extension and enhancement of the iDyn library
 - The wholeBodyDynamics software module

Simulink wrapper



- "**Rapid prototyping**" of controllers is a desired feature which speeds up final implementation on the real platform.
- Single additional dependence: Simulink coder.
- Exploitation of **Simulink** and **MATLAB** toolboxes, e.g. System Identification, Robust Control, Model Predictive Control, Optimization, Neural Networks, Simulink Control Design, Stateflow.
- Higher level of **abstraction** to the Whole-Body and Sensors Interfaces.

Model Interface in Simulink



Building C/C++ MEX Files



ySynchronizer

Simulink allows to easily create control thread. Synchronization (real-time or with Gazebo) can be achieved with a proper Simulink block.

MATLAB wrapper

robotology-playground / **mex-wholebodymodel**

Unwatch 28 ⚡ Star 0 Fork 0

Matlab MEX interface to the iWholeBodyModel interface. — Edit

43 commits 3 branches 0 releases 3 contributors

branch: master mex-wholebodymodel / +

Code Issues Pull Requests

```
1
2
3 - wholeBodyModel('model-initialise');
4
5 - qj = rand(25,1);
6 - xTb = wholeBodyModel('forward-kinematics',qj,'l_gripper');
7
8 %Setting State to random values
9 - qj = rand(25,1);dqj = rand(25,1);dxb = rand(6,1);
10
11 %Mex-WholeBodyModel Components
12 - M = wholeBodyModel('mass-matrix',qj);
13 - H = wholeBodyModel('generalised-forces',qj,dqj,dxb);
14 - DjDq1 = wholeBodyModel('djdq',qj,dqj,dxb,refLink1);
15 - DjDq2 = wholeBodyModel('djdq',qj,dqj,dxb,refLink2);
16 - J1 = wholeBodyModel('jacobian',qj,refLink1);J2 = wholeBodyModel('jacobian',qj,refLink2);
17 - J3 = wholeBodyModel('jacobian',qj,refLink1);J4 = wholeBodyModel('jacobian',qj,refLink2);
```

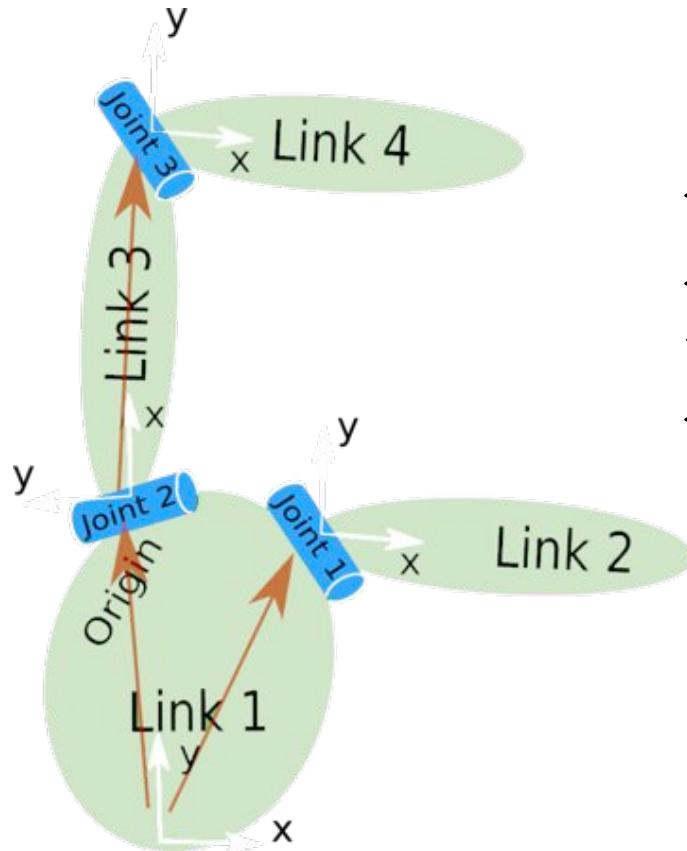
Outline

- T1.1 Software architecture:
 - The WBI (wholeBodyInterface)
 - The WBIToolbox (simulink interface to the WBI)
- T1.2 Simulator:
 - The YARP interface to Gazebo
(gazebo_yarp_plugin)
- T1.4 System dynamics estimation software:
 - The iDynTree::computeRegressor() function
- T1.5 Extension and enhancement of the iDyn library
 - The wholeBodyDynamics software module

Unified Robot Description Format

- XML format for representing a robot model.
- Kinematic and dynamic description of the robot.
- Tree-based representation (no parallel configurations).
- Rigid links (no flexible elements).
- Models different elements: e.g. links, joints, transmissions and visual and collision properties.

URDF: example



```
<robot name="test_robot">
  <link name="link1">
    <inertial> <mass value="10"/> <inertia
      ixx="0.4" ixy="0.0" ixz="0.0"
      iyy="0.4" iyz="0.0" izz="0.2"/></inertial>
  </link>
  <link name="link2" />
  <link name="link3" />

  <joint name="joint1" type="continuous">
    <parent link="link1"/>
    <child link="link2"/>
    <origin xyz="5 3 0" rpy="0 0 0" />
    <axis xyz="-0.9 0.15 0" />
  </joint>
  ....
</robot>
```

URDF: Benefits

- Use of the same model for both simulation (e.g. Gazebo) and control (e.g. by using KDL).
- Simple to understand and use.
- Widely used and supported.
- Under active development.

Gazebo Simulator

- Modular Physic engine (ODE, Bullet, Simbody).
- Dynamic model and meshes from URDF file.
- Independent from ROS.
- APIs allows custom interfaces.

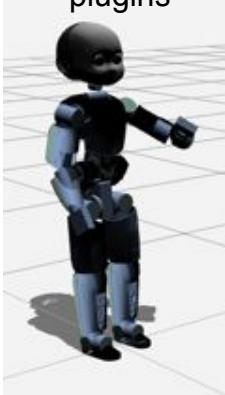


Gazebo - YARP plugin

- Allows to interact with the simulator via YARP Devices.
- One YARP device correspond to a plugin in Gazebo (e.g. Force/Torque device is a Gazebo sensor plugin).
- Switching between simulator and real robot with no code change.

Gazebo-YARP Architecture

YARP devices
talking to
Gazebo via
gazebo-
plugins



ROBOT_NAME
icubGazeboSim

YARP devices
talking to real
hardware



ROBOT_NAME
cub



Control Board

/ROBOT_NAME/right_leg/rpc:i
/ROBOT_NAME/right_leg/state:o
/ROBOT_NAME/left_leg/rpc:i
/ROBOT_NAME/left_leg/command:i
/ROBOT_NAME/left_leg/state:o
...
/ROBOT_NAME/right_leg/state:o

F/T Sensors

/ROBOT_NAME/left_leg/analog:o
/ROBOT_NAME/left_leg/analog:o
...
/ROBOT_NAME/right_leg/analog:o

IMU

/ROBOT_NAME/inertia

User module

/MY_MODULE_NAME/rpc:o
/MY_MODULE_NAME/command:o
/MY_MODULE_NAME/state:i

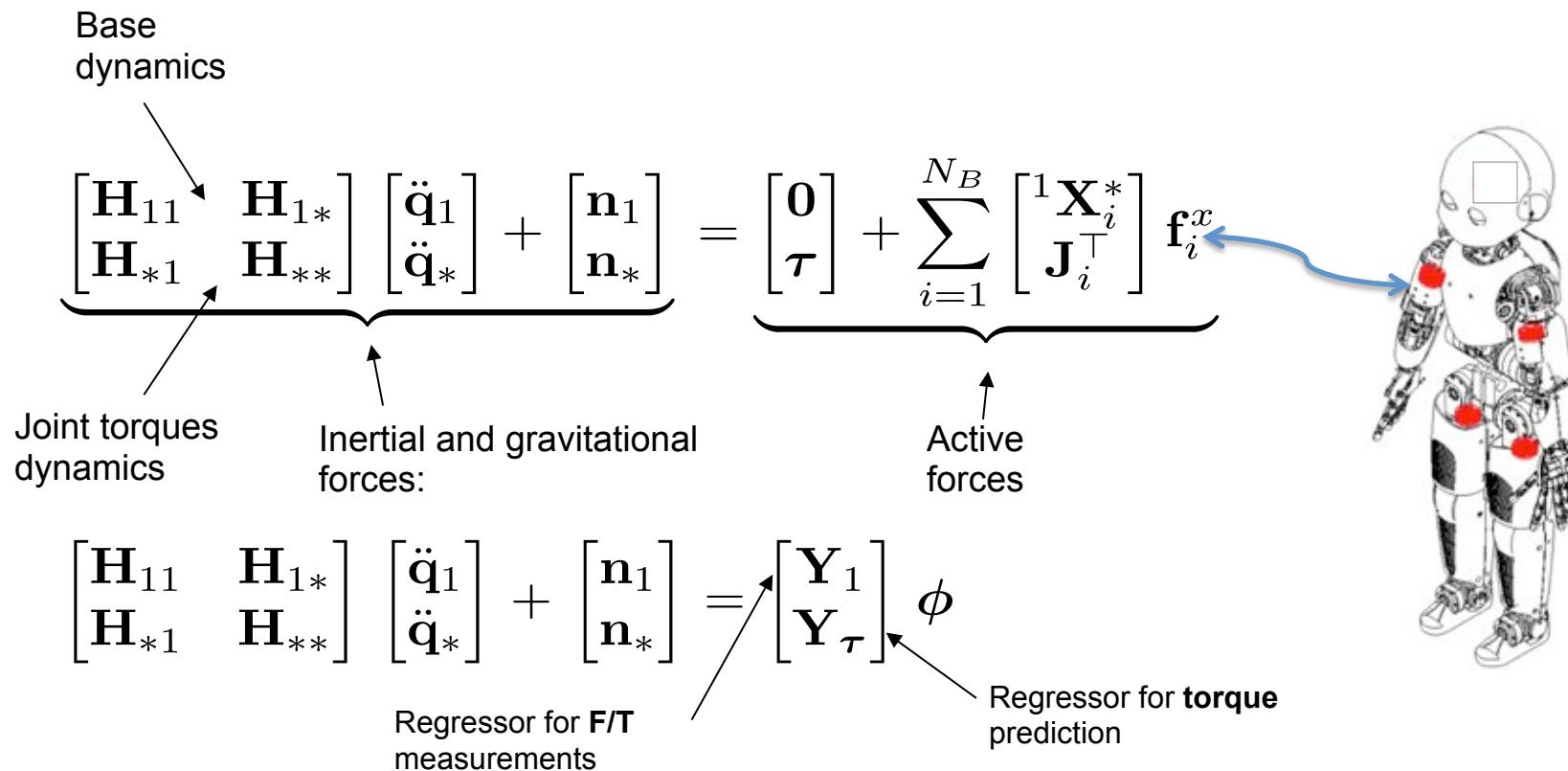
Outline

- **T1.1 Software architecture:**
 - The WBI (wholeBodyInterface)
 - The WBIToolbox (simulink interface to the WBI)
- **T1.2 Simulator:**
 - The YARP interface to Gazebo
(gazebo_yarp_plugin)
- **T1.4 System dynamics estimation software:**
 - The DynamicRegressorGenerator class
- **T1.5 Extension and enhancement of the iDyn library**
 - The wholeBodyDynamics software module

The DynamicRegressorGenerator class

- Numerical computation of identifiable dynamic parameters subspace (so called “base parameters”).
- Unique open-source C++ module for estimating “base parameters”.
- Implemented on a CoDyCo module named staticInertialIdentification.
- Estimates “base parameters” with a classical on-line least square solution.

The DynamicRegressorGenerator class



Regressors of other quantities

- Mass

$$m = \mathbf{Y}_m \phi$$

- First Moment of Mass

$$m^f \mathbf{c} = {}^f \mathbf{Y}_{mc} \phi$$

- Momentum

$$E_{t_1} - E_{t_0} = \mathbf{Y}_{\Delta E} (\mathbf{q}_{t_1}, \dot{\mathbf{q}}_{t_1}, \mathbf{q}_{t_0}, \dot{\mathbf{q}}_{t_0}) \phi$$

- Energy
-

Identifiable subspace

Problem: only a subspace of the \mathbb{R}^{10N_B} space of the inertial parameters can be identified using the dynamic regressor \mathbf{Y} .

This “identifiable subspace”: $I_{\mathbf{Y}}$.

Mathematical definition:

$$N_{\mathbf{Y}} = \{\boldsymbol{\phi} \in \mathbb{R}^{10N_B} : \mathbf{Y}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\boldsymbol{\phi} = \mathbf{0} \quad \forall \mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}\}$$

$$I_{\mathbf{Y}} = N_{\mathbf{Y}}^{\perp}$$

Identifiable subspaces results

Using a mix of results from literature and new demonstration, it is possible to prove:

$$I_{\mathbf{Y}} = I_{\mathbf{Y}_1} = I_{\mathbf{Y}_{\Delta E}} = I_{\mathbf{Y}_h}$$

$$I_{\mathbf{Y}_\tau} \subseteq I_{\mathbf{Y}_1} \quad I_{\mathbf{Y}_{\tau_f}} \subseteq I_{\mathbf{Y}_1} \quad I_{\mathbf{Y}_m} \subseteq I_{\mathbf{Y}_1} \quad I_{\mathbf{Y}_{m\mathbf{c}}} \subseteq I_{\mathbf{Y}_1}$$

The parameters estimated from \mathbf{Y}_1 or $\mathbf{Y}_{\Delta E}$ can be used to calculate all the other quantities .

Outline

- T1.1 Software architecture:
 - The WBI (wholeBodyInterface)
 - The WBIToolbox (simulink interface to the WBI)
- T1.2 Simulator:
 - The YARP interface to Gazebo
(gazebo_yarp_plugin)
- T1.4 System dynamics estimation software:
 - The DynTree::getDynamicsRegressor() function
- T1.5 Extension and enhancement of the iDyn library
 - The wholeBodyDynamicsTree software module

iDyn enhancement

- iDyn reimplemented in iDynTree. Software structure is now fully compatible with (Fumgalli et al., 2012).
- Computes whole-body dynamics using the KDL basic functionalities (recursice Newton-Euler).
- Load URDF robot file description, therefore fully compatible with the gazebo Gazebo (i.e. icub=icubGazeboSim).

iDynTree structure

