ISTITUTO ITALIANO
DI TECNOLOGIA

iDynTree :
Free Floating Dynamics Library

Silvio Traversaro

iCub Facility

Istituto Italiano di Tecnologia, Genova, Italy

Loading model

Models can be built programmatically but typically are loaded from **URDF** files (the same robot description format used in **ROS**).

As sometimes we are not interested in all joints contained in a robot, we can explicitly specify the **joints** that we want to load, and their order, using the **iDynTree::ModelLoader** class.

From this class, you obtain a **iDynTree::Model** that contain the structure and the parameters of the model, that you can pass to **iDynTree::KinDynComputations** that

```
ModelLoader mdlLoader;

std::vector<std::string> joints; // specify the used joints
joints.push_back(“torso_pitch”);
...
joints.push_back(“r_wrist_pitch”);

mdlLoader.loadReducedModelFromFile(“./model.urdf”, joints);

Model model = mdlLoader.model();

KinDynComputations kinDynComp;
kinDynComp.loadRobotModel(mdlLoader.model());

// You can now use the kinDynComp object to compute terms
// of the dynamics equations, transformation between
// frames, jacobians
```

Caveat on iDynTree <--> YARP integration

- Conversion utilities between **YARP** and **iDynTree** are available in the `<iDynTree/yarp/YARPConversions.h>` header.

- **iDynTree** uses *radians* for all its interfaces, **YARP** typically uses *degrees*.

- **iDynTree** assume in input and in output always *floating base quantities*, you will need to do to the appropriate conversions if you want to use them for **fixed base robots** control.

- See the **impedance_control-tutorial** for more on this!