Introduction to iDyn

Serena Ivaldi
Matteo Fumagalli
Marco Randazzo
Ugo Pattacini
Francesco Nori

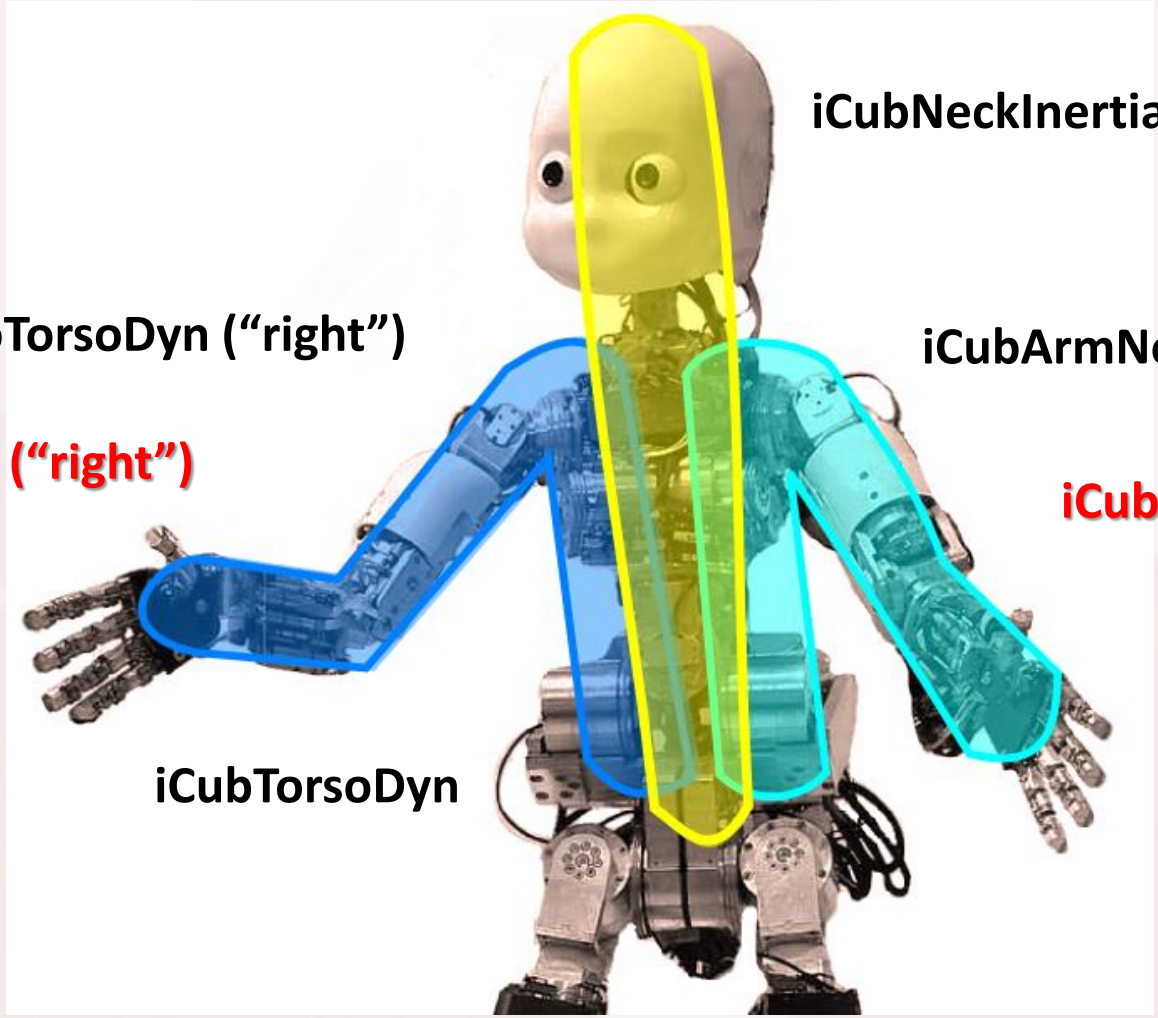
What you can get from iDyn :)

Library overview

- **iDyn core**: links, chains, limbs... and iCubLimbs of course
- **RecursiveNewtonEuler** : the math underneath force/torque computations in a chain using Newton-Euler recursive formulation
- **iDynInv** : computation of the inverse dynamics using the RNE
- **iDynSensor** : computation of forces and torques in a chain given single FT sensor measurements
- **iDynBody** : whole body dynamics, i.e. interconnection of multiple chains
- **iDynTransform**: projection of forces along a chain

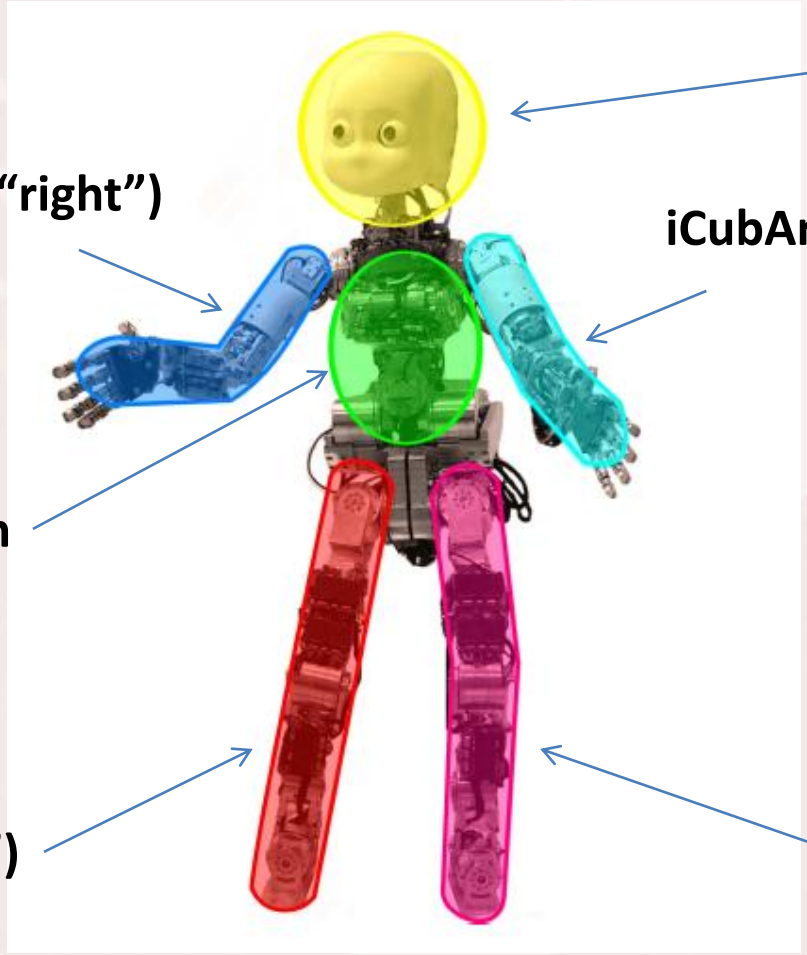
Note: iDyn is a generic library (for any robot) ..
… but of course it already has everything specific for iCub!!!

iCub limbs



compatibility with iKin!

iCub limbs



iCubNeckInertialDyn

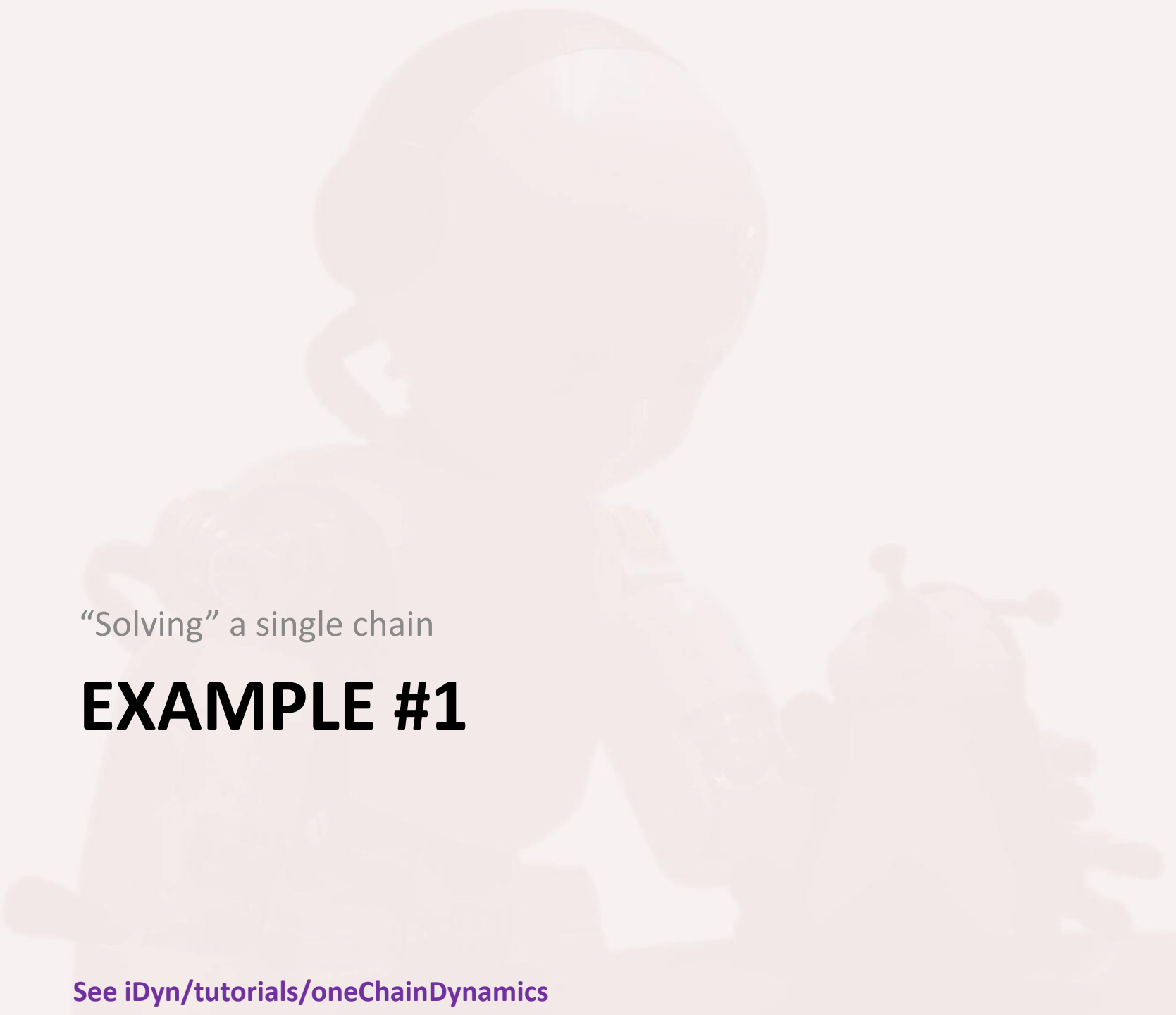
iCubArmNoTorsoDyn (“right”)

iCubArmNoTorsoDyn (“left”)

iCubTorsoDyn

iCubLegDyn (“right”)

iCubLegDyn (“left”)

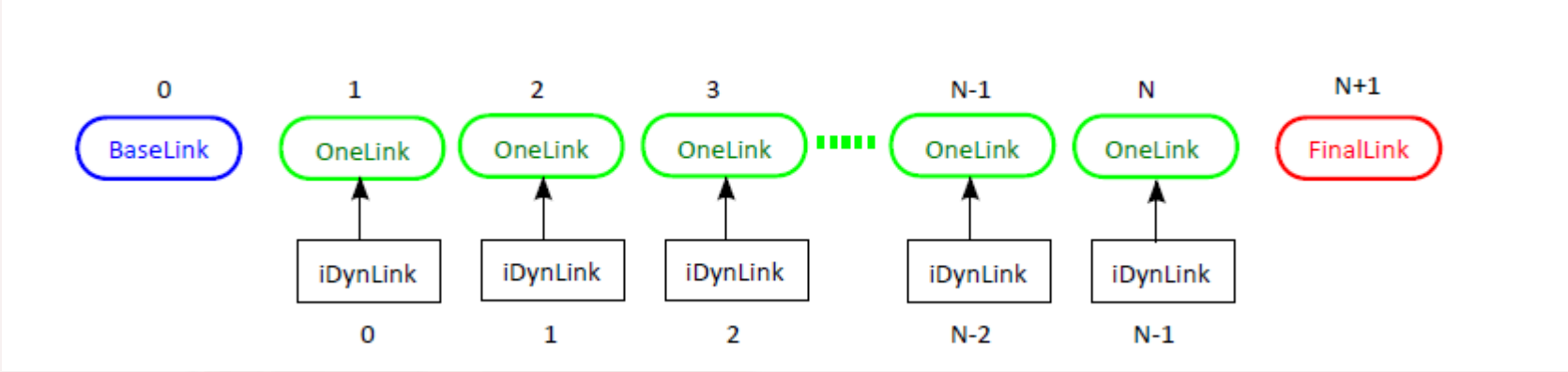


"Solving" a single chain

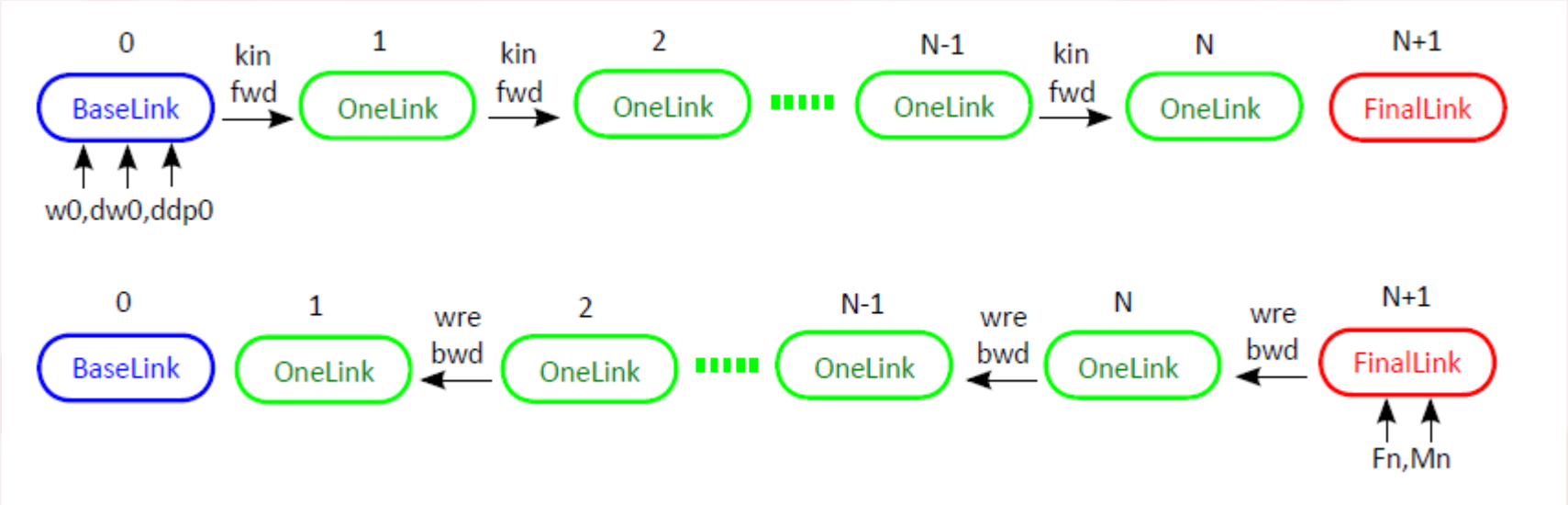
EXAMPLE #1

See iDyn/tutorials/oneChainDynamics

Solving a single chain



Newton- Euler recursive formulation



```
#include <iCub/iDyn/iDyn.h>

using namespace std;
using namespace yarp::sig;
using namespace iCub::iDyn;

iCubArmDyn   armTorsoDyn("right");

armTorsoDyn.releaseLink(0);
armTorsoDyn.releaseLink(1);
armTorsoDyn.releaseLink(2);

armTorsoDyn.setAng(q);
armTorsoDyn.setDAng(dq);
armTorsoDyn.setD2Ang(ddq);

w0=dw0=ddp0=0.0;
ddp0[2]=9.81;

Fend = Muend = 0.0;
```

how to include iDyn

arm + torso as iKin

unblock torso

set joint position, velocity and acceleration

Init kinematic phase: w0, dw0, ddp0

Init wrench phase: F & Mu at the end-eff.

Before any dynamic computation, call prepare() method!

```
armTorsoDyn.prepareNewtonEuler(DYNAMIC);
```

1) Init Kinematics in Base
2) Solve Kinematics Forward
3) Init Wrench at the End
4) Solve Wrench Backward

```
armTorsoDyn.computeNewtonEuler(w0,dw0,ddp0,Fend,Muend);
```

Finally retrieve your result :)

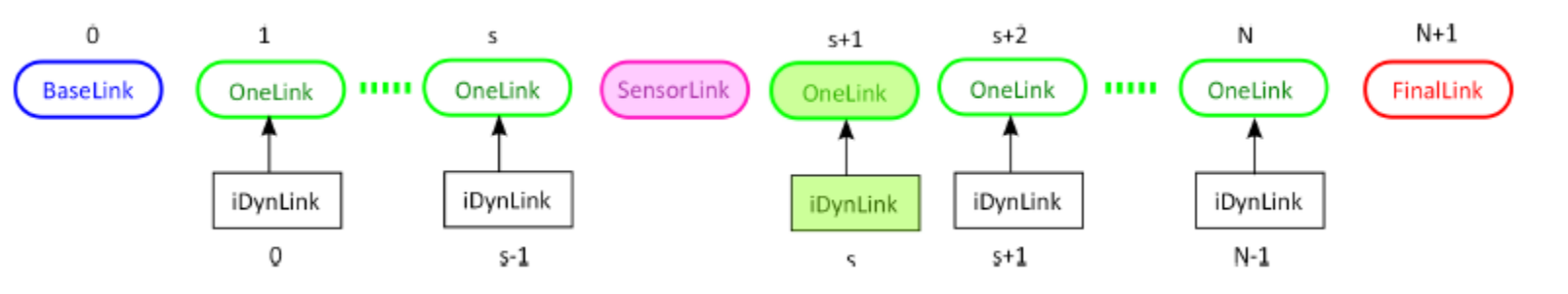
```
Matrix F = armTorsoDyn.getForces();
Matrix Mu = armTorsoDyn.getMoments();
Vector Tau = armTorsoDyn.getTorques();
```

"Solving" a single chain having a FT sensor inside..

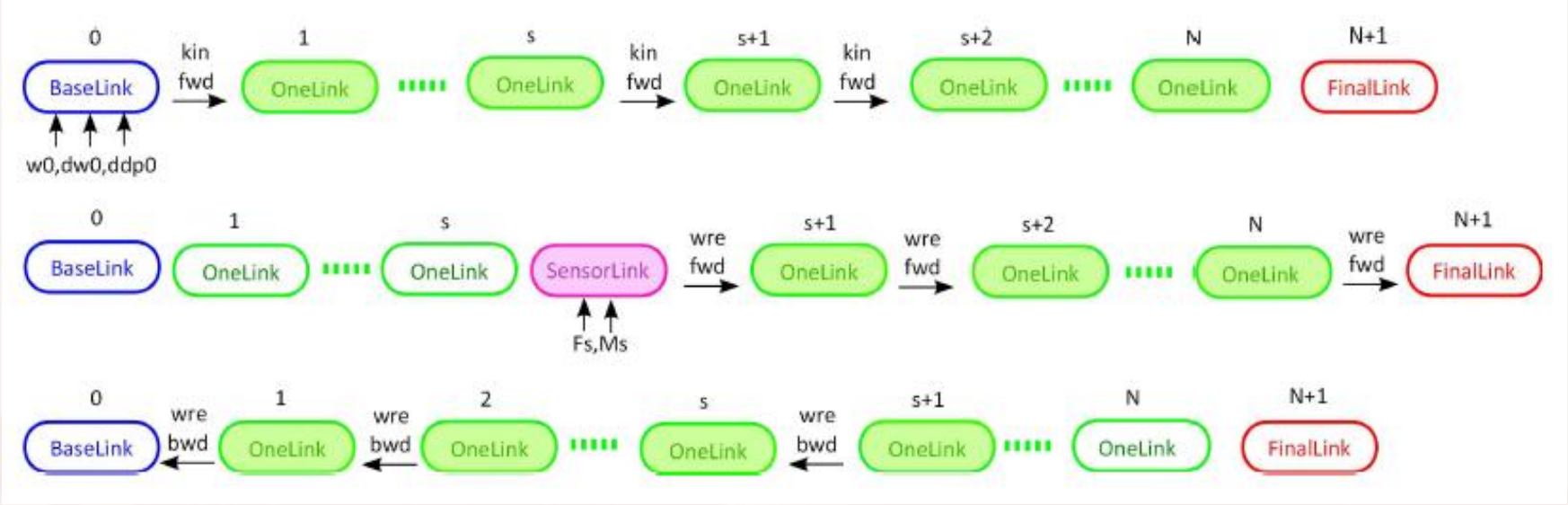
EXAMPLE #2

See iDyn/tutorials/oneChainWithSensor

Solving a single chain with a FT sensor



Newton- Euler recursive formulation: wrench phase is “split”



```
#include <iCub/iDyn/iDyn.h>
#include <iCub/iDyn/iDynInv.h>
using namespace iCub::iDyn;

iCubArmNoTorsoDyn *arm = new iCubArmNoTorsoDyn("right");

iDynSensorArmNoTorso *armWSensorSolver
= new iDynSensorArmNoTorso(arm,STATIC,VERBOSE);

arm->setAng(q);  arm->setDAng(dq);  arm->setD2Ang(ddq);

arm->prepareNewtonEuler(STATIC);

w0=dw0=ddp0=0.0; ddp0[2]=9.81;
arm->initKinematicNewtonEuler(w0,dw0,ddp0);

armWSensorSolver->computeFromSensorNewtonEuler(Fsens,Musens);

Matrix F = arm->getForces();
Matrix Mu = arm->getMoments();
Vector Tau = arm->getTorques();
```

iDynInv inclusion

Arm only!

The sensor type is automatically selected from the arm type (also its properties)

As usual…

Remember to call prepare()! STATIC, DYNAMIC…

Set kinematic data on the base

1) Solve Kinematics phase forward
2) From sensor to base: wrench backward
3) From sensor to end: wrench forward

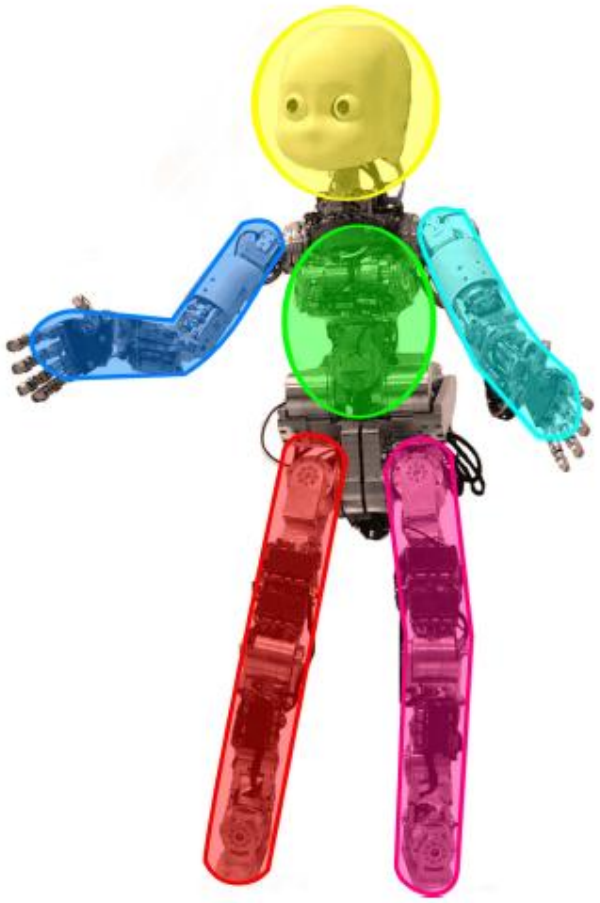
Finally retrieve your result :)

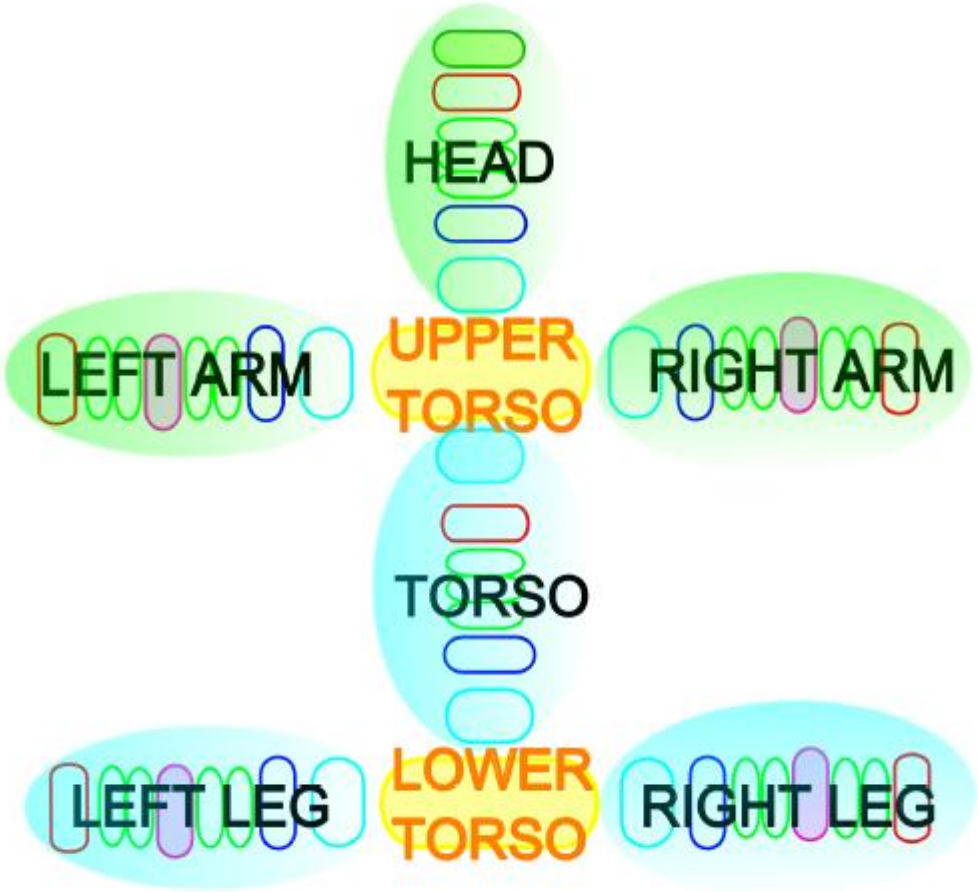
Playing with iCubWholeBody.. Estimation of FT sensor measurements

EXAMPLE #3

See iDyn/tutorials/wholeBody_FTSensor_estimate

iCub whole Body





```
#include <iCub/iDyn/iDynBody.h>
using namespace iCub::iDyn;
```

Usual iDyn inclusion

```
w0=dw0=ddp0=0.0; ddp0[2]=9.81;
```

Inertial sensor measurements!!!

```
icub.upperTorso->setAng("head",q_head);
icub.upperTorso->setAng("right_arm",q_rarm);
icub.upperTorso->setAng("left_arm",q_larm);
icub.upperTorso->setDAng("head",q_head);
icub.upperTorso->setDAng("right_arm",q_rarm);
icub.upperTorso->setDAng("left_arm",q_larm);
icub.upperTorso->setD2Ang("head",q_head);
icub.upperTorso->setD2Ang("right_arm",q_rarm);
icub.upperTorso->setD2Ang("left_arm",q_larm);
```

Setting joints position, velocity, acceleration… for the whole body!!!

Upper Torso: head + left/right arm

```
icub.lowerTorso->setAng("torso",q_torso);
icub.lowerTorso->setAng("right_leg",q_rleg);
icub.lowerTorso->setAng("left_leg",q_lleg);
icub.lowerTorso->setDAng("torso",q_torso);
icub.lowerTorso->setDAng("right_leg",q_rleg);
icub.lowerTorso->setDAng("left_leg",q_lleg);
icub.lowerTorso->setD2Ang("torso",q_torso);
icub.lowerTorso->setD2Ang("right_leg",q_rleg);
icub.lowerTorso->setD2Ang("left_leg",q_lleg);
```

LowerTorso: torso + left/right leg

icub.upperTorso->setInertialMeasure(w0,dw0,ddp0);

Set the inertial sensor measurements on the head

Matrix fm_sens_up =
icub.**upperTorso**->estimateSensorsWrench(FM_up);

1) Solve kinematics and wrench in the head
2) Propagate kinematics through node to the arms
3) Solve wrench in the arms
4) Find FT sensor wrench

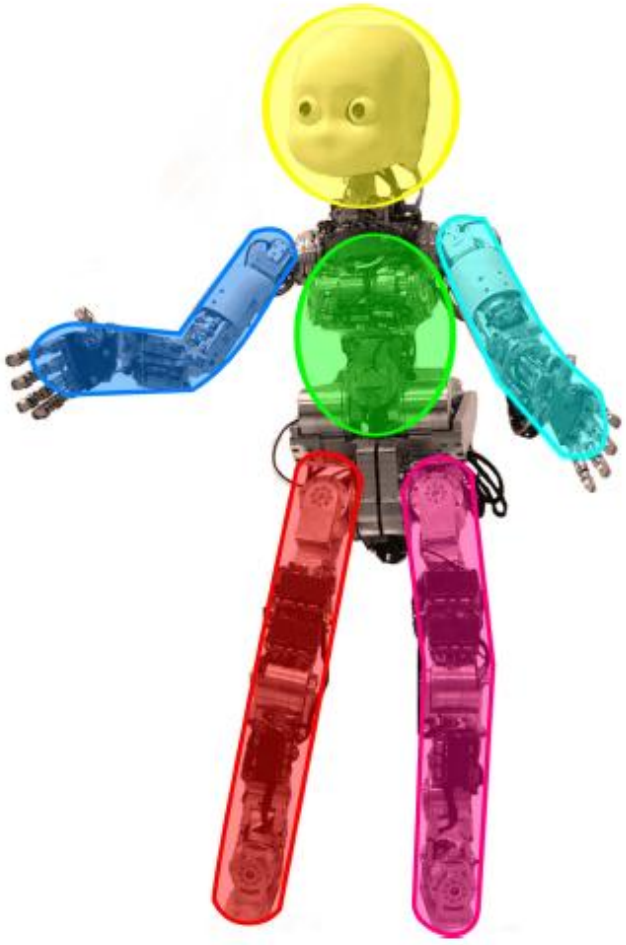
icub.**attachTorso**();

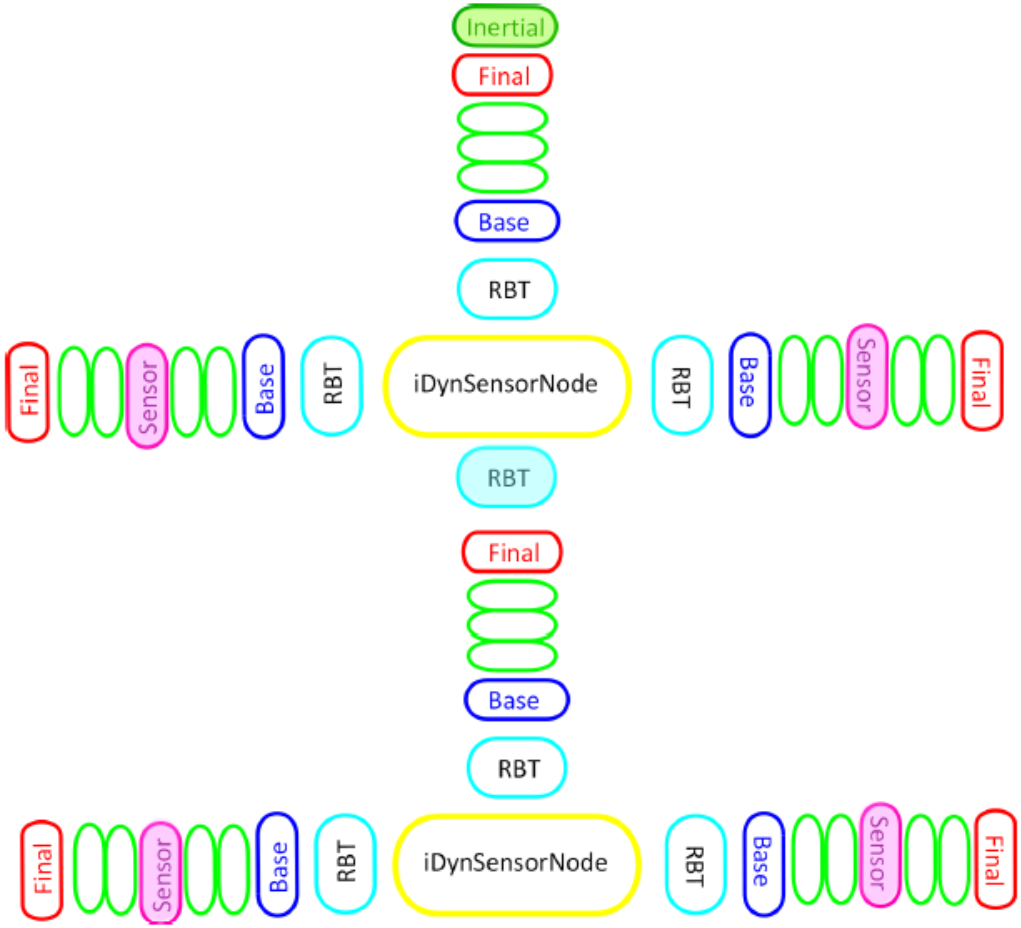
Kinematic and wrench data are set from UpperTorso node to Torso limb

1) Solve kinematics and wrench in the torso
2) Propagate kinematics through node to the legs
3) Solve wrench in the legs
4) Find FT sensor wrench

Matrix fm_sens_lo =
icub.**lowerTorso**->estimateSensorsWrench(FM_lo,NODE_AFTER_ATTACH);

iCub whole Body





… for more details..

Dynamic equations:

Control of robotic manipulators

Handbook of Robotics

Code documentation:

http://eris.liralab.it/iCub/main/dox/html/group__iDyn.html

Code:

iCub/main/src/libraries/iDyn

Coming soon …

High-performance
(@PC104 level)

Impedance control !!