Whole-body Compliant Dynamical Contacts in Cognitive Humanoids

Whole-body Compliant Dynamical Contacts in Cognitive Humanoids

proj. no. 600716

WP1: Systems Integration, Standardization and Evaluation on the iCub robot

Contributor: IIT

Fondazione Istituto Italiano di Tecnologia.

Robotics, Brain and Cognitive sciences Department.

Outline

* T1.1 Software architecture:
  - The wholeBodyInterface and the WBIToolbox
* T1.4 System dynamics estimation software:
  - Force/torque sensor calibration.
* T1.5 Extension and enhancement of the iDyn library
  - The maximum-a-posteriori dynamics.

SEVENTH FRAMEWORK PROGRAMME

CODYCO

WBI-Toolbox in a nutshell

Simulink Toolbox wrapping a YARP-based implementation of the **Whole-Body Interface** (WBI) C++ library.

MEX-Files
dynamically linked C/C++ code and libraries

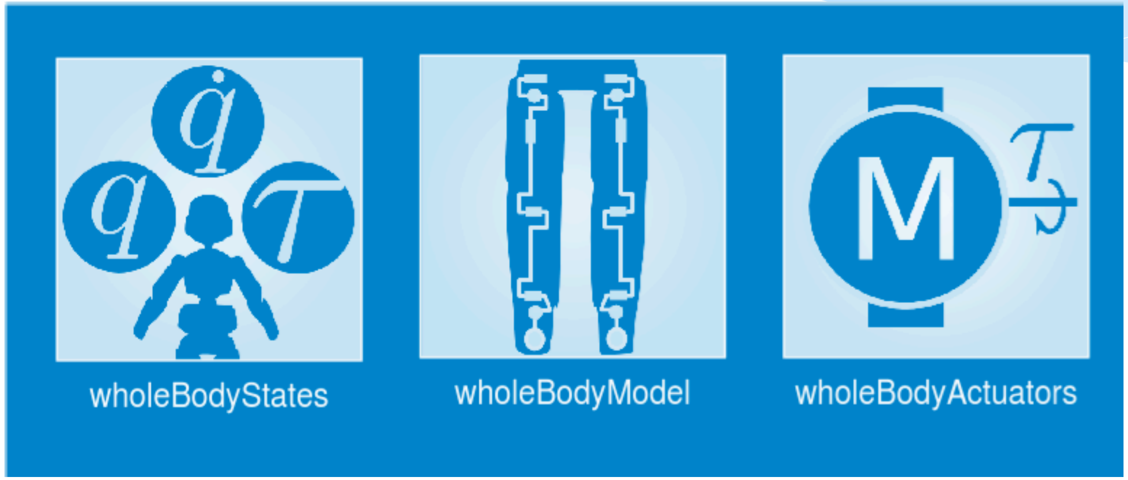
❯

MATLAB Interpreter
load and executes

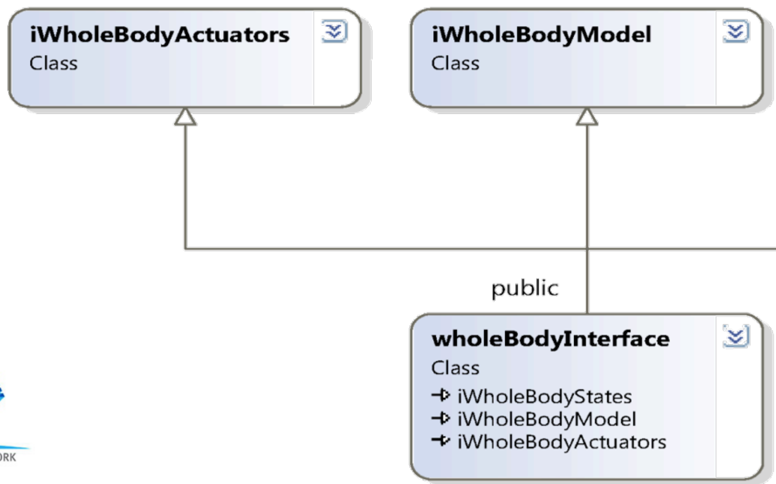
❯

SIMULINK
S-function

WBI-Toolbox in a nutshell



(Simulink) WBI-Toolbox:
a mex library linking against
a wholeBodyInterface
implmentation

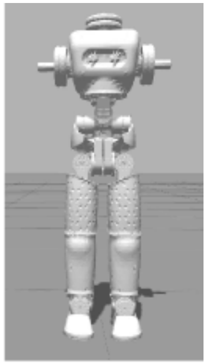


(C++) wholeBodyInterface:
a software abstraction layer
for whole-body motion
control

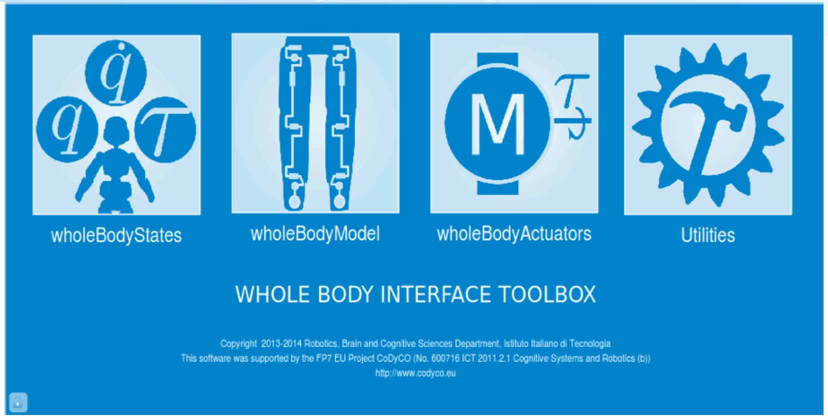
WBI-Toolbox overview

iCub

COMAN



iCub
(Heidelberg)



- Simulink toolbox for "Rapid prototyping" of controllers.
- Works with yarp-based robots (e.g. iCub, COMAN)
- (Even) Higher level of abstraction for the Whole Body Interface.
- Better alternative to the use of YARP JAVA bindings.

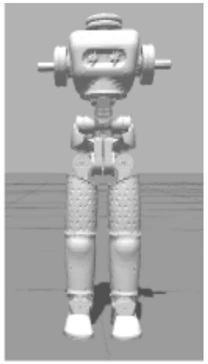
SEVENTH FRAMEWORK PROGRAMME

CODYCO

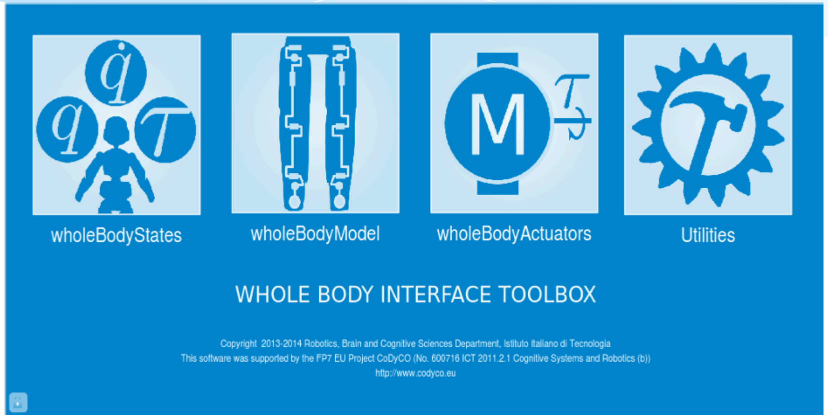
WBI-Toolbox overview

iCub

COMAN



iCub
(Heidelberg)



- Direct exploitation of Simulink and MATLAB toolboxes.
- Faster approach to the real robot.
- Easier debugging of control issues.
- Synchronization with Gazebo via Gazebo clock plugin.
- Supports Linux, Mac OS X and Windows (under revision).

SEVENTH FRAMEWORK PROGRAMME

WBI-Toolbox dependencies

- **YARP** - Robotics middleware
- **iCub** - Not strictly necessary
- **CoDyCo**
  - **iDynTree** Library
  - **wholeBodyInterface** Library

- Matlab (R2012a+)
- Simulink
- Simulink Coder - Toolbox

CODYCO SUPERBUILD

SEVENTH FRAMEWORK PROGRAMME

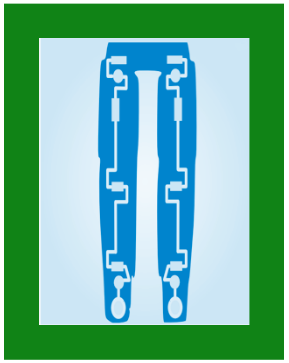
CODYCO

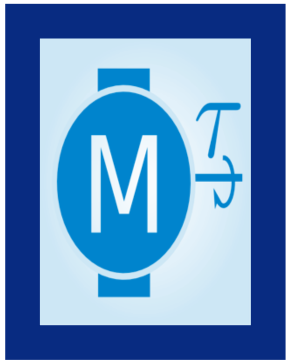
WBI-Toolbox example

Jorhabib Eljaik et al.

Whole Body Impedance Controller quickly implemented on Simullink

$$-K_p(q_j - q_{j_0}) - K_d\dot{q}_j + g = \tau_j$$



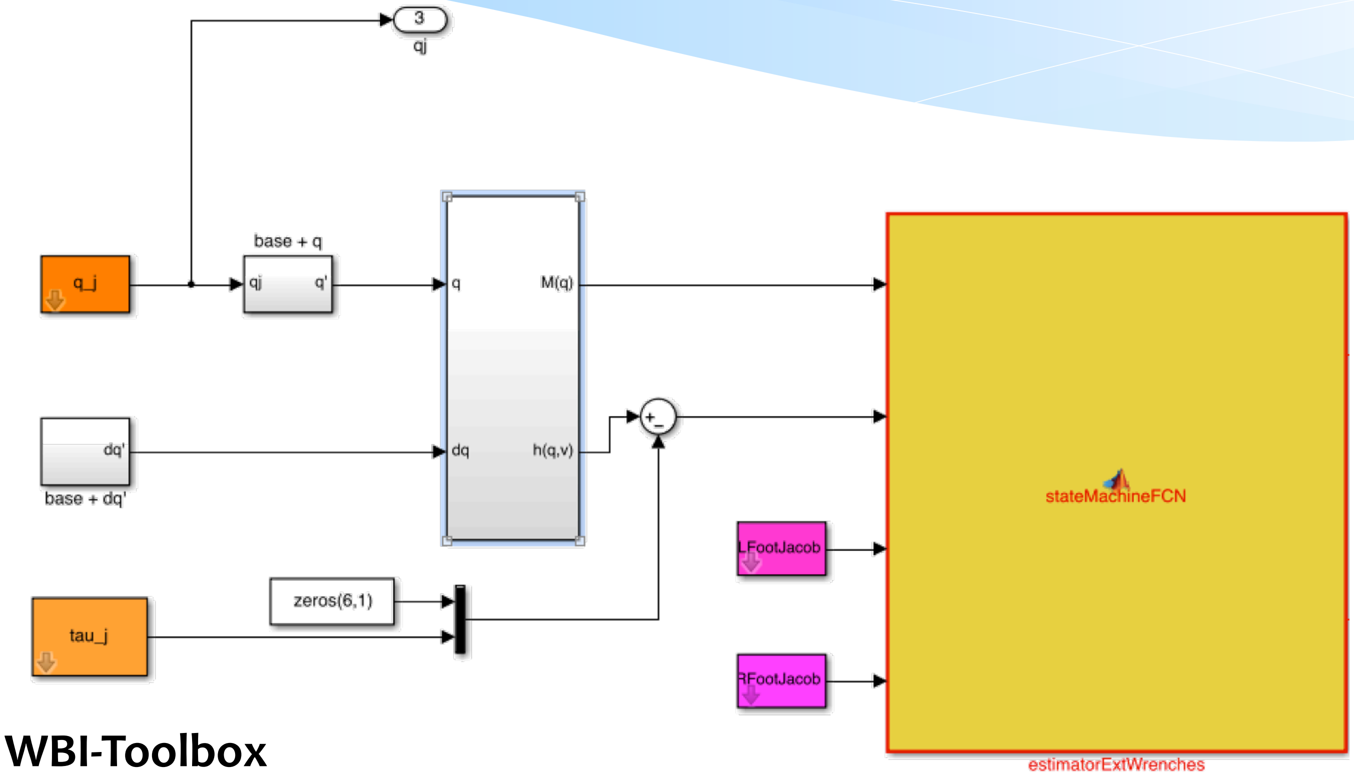


SEVENTH FRAMEWORK PROGRAMME

CODYCO

WBI-ToolboxControllers

Daniele Pucci et al.

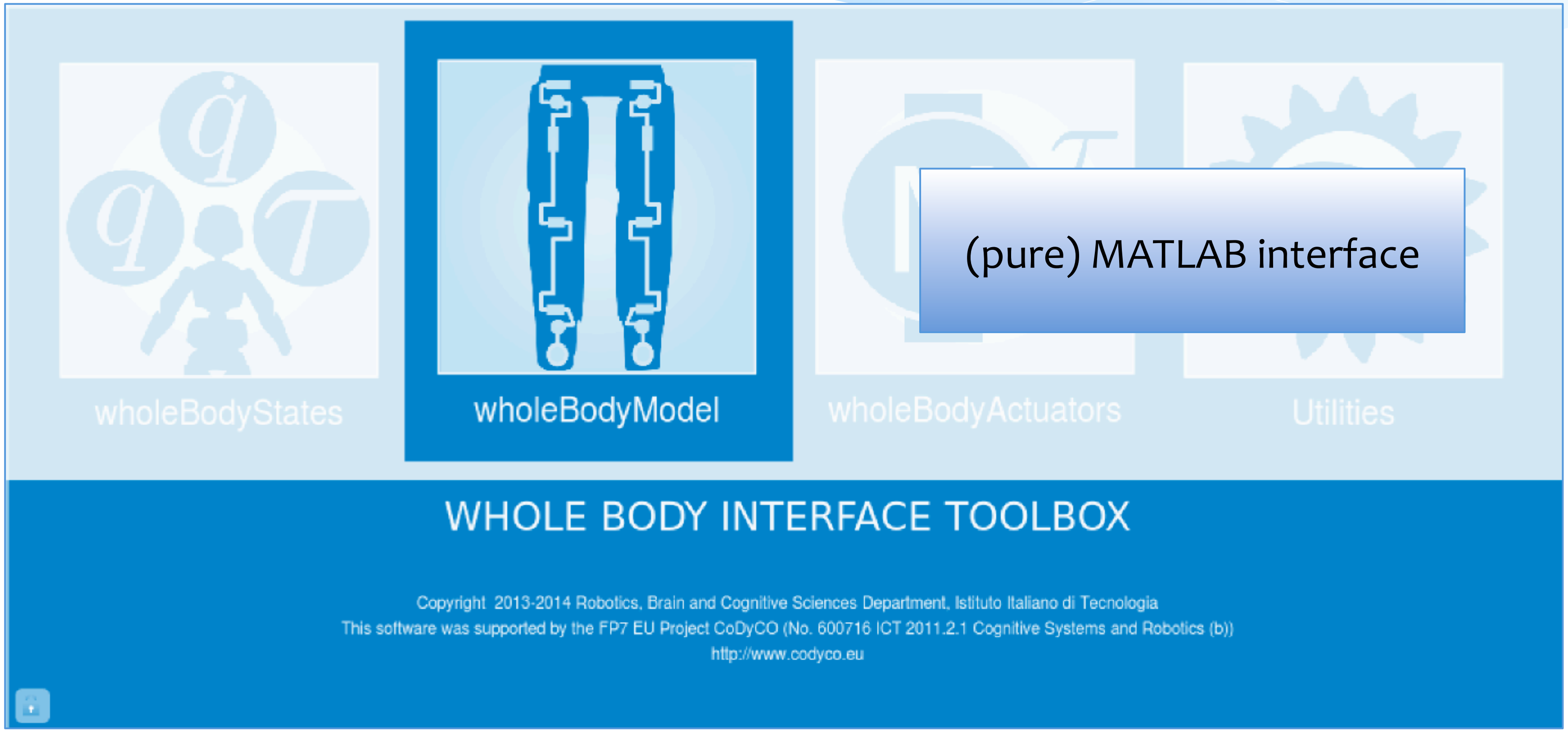


WBI-Toolbox
https://github.com/robotology-playground/WBI-Toolbox

WBIToolboxControllers
https://github.com/robotology-playground/WBI-ToolboxControllers

mexWholeBodyModel

Naveen Kuppuswamy et al.



(pure) MATLAB interface

mex-wholebodymodel
https://github.com/robotology-playground/mex-wholebodymodel

mexWholeBodyModel

Naveen Kuppuswamy et al.

EXAMPLE: accessing the kinematics, mass-matrix and Jacobians

```
%%
wholeBodyModel('model-initialise');
link = 'l_hip_1';
x    = wholeBodyModel('forward-kinematics',qj,link);
M    = wholeBodyModel('mass-matrix',qj);
h    = wholeBodyModel('generalised-forces',qj,dqj,dxb);
dJdq = wholeBodyModel('djdq',qj,dqj,dxb,link);
J    = wholeBodyModel('jacobian',qj,link);
```

Source code and installation instructions

mex-wholebodymodel

https://github.com/robotology-playground/mex-wholebodymodel

Outline

* T1.1 Software architecture:
  - The wholeBodyInterface and the WBIToolbox
* T1.4 System dynamics estimation software:
  - Force/torque sensor calibration.
* T1.5 Extension and enhancement of the iDyn library
  - The maximum-a-posteriori dynamics.

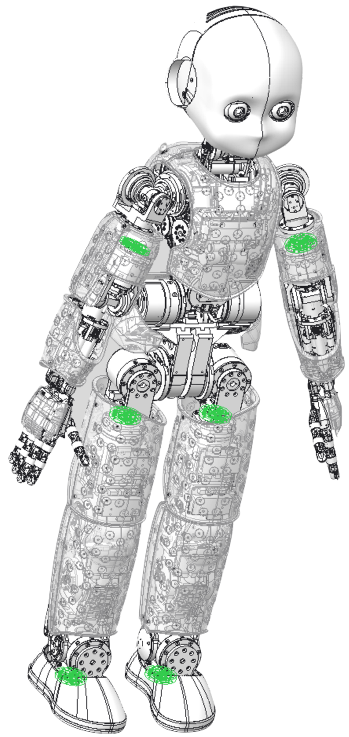
Force/torque sensor calibration: problem statement

Motivations:

- Force/Torque sensors require periodic recalibration.
- Calibration should be performed in-situ.

Problem:

- Estimate the calibration matrix from raw measurements using calibrated accelerometers.



SEVENTH FRAMEWORK PROGRAMME

CODYCO

Force/torque sensor calibration: problem formalization

Model:

* We measure the force/torque to counterbalance gravity at different orientations of a given rigid body.

Measurements:

* Raw strain-gauges measurements are a linear affine transformation of the applied force/torque.
* Accelerometers measure the proper acceleration expressed in body coordinates.

Assumptions:

* Measurements are taken in static conditions (null velocities).
* No knowledge of the rigid body dynamic parameters but known calibration masses are added.

Force/torque sensor calibration: problem formalization

Calibration matrix Raw strains

Force/torque Offset

$$w = C(r + o)$$

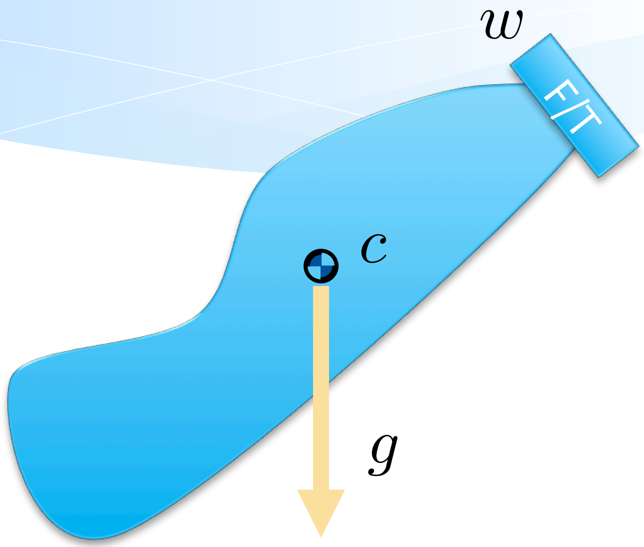
Rotation matrix

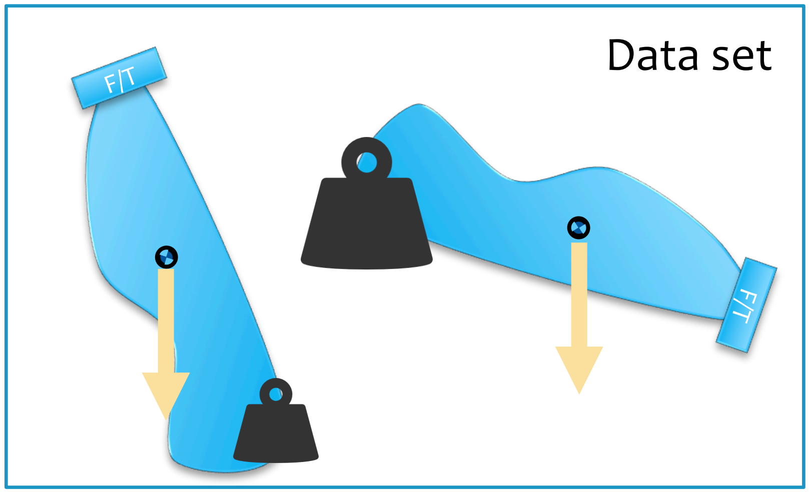
Gravity (constant) Gravity (body)

$$\bar{g} = T g$$

$$\begin{aligned} w &= M(c)g, \\ M(c) &= m \begin{pmatrix} I_3 \\ c\times \end{pmatrix} \end{aligned}$$

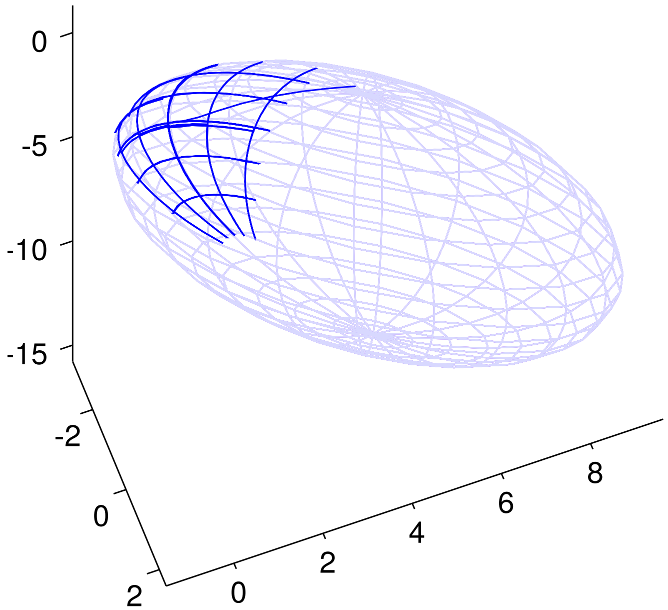
Center of mass (body) Mass





Force/torque sensor calibration: problem structure

* **Result 1:** in the noise-free case, raw strain-gauges measurements lie on a three dimensional ellipsoid.



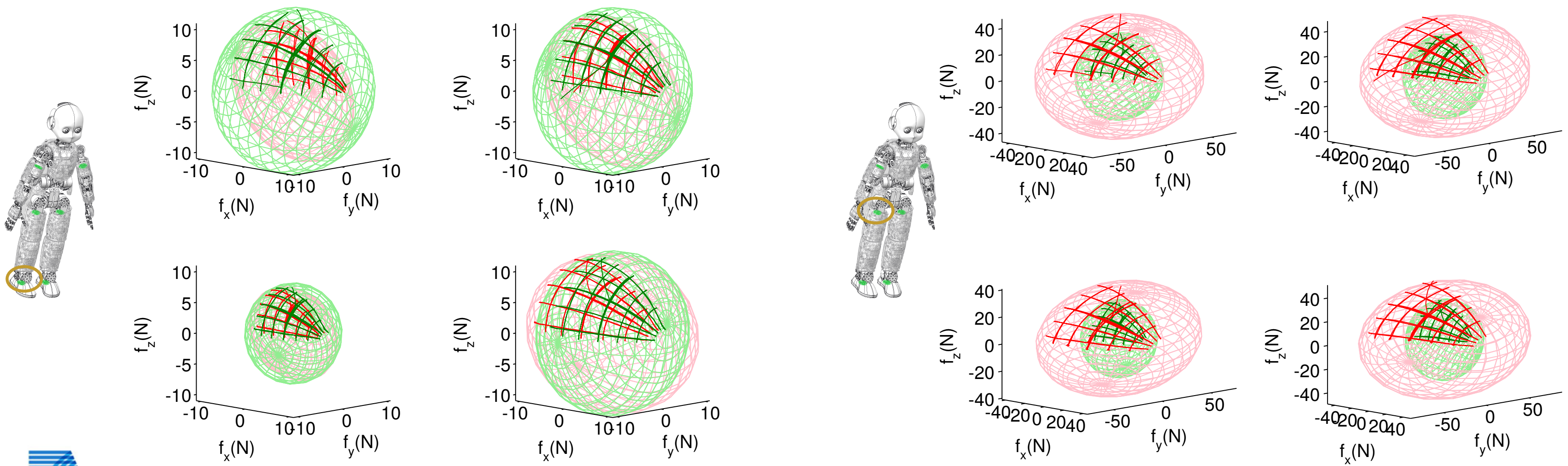
Force/torque sensor calibration: problem solution

* **Result 2.1:** offset on raw measurements can be estimated from data using a linear least-square procedure (i.e. the computation of a pseudo-inverse).

* **Result 2.2:** after removing the offset, the calibration matrix, the mass and the center of mass position can be estimated with a second least-square procedure.

Force/torque sensor calibration: experimental results

* **Result 3:** after calibration on a training set, calibrated force and torques lie on a sphere (test set).



Open-source code: https://github.com/robotology-playground/insitu-ft-calibration

S. Traversaro et al. (*ICRA 2015*)

Outline

* T1.1 Software architecture:
  - The wholeBodyInterface and the WBIToolbox
* T1.4 System dynamics estimation software:
  - Force/torque sensor calibration.
* T1.5 Extension and enhancement of the iDyn library
  - The maximum-a-posteriori dynamics.

Maximum-a-posteriori dynamics

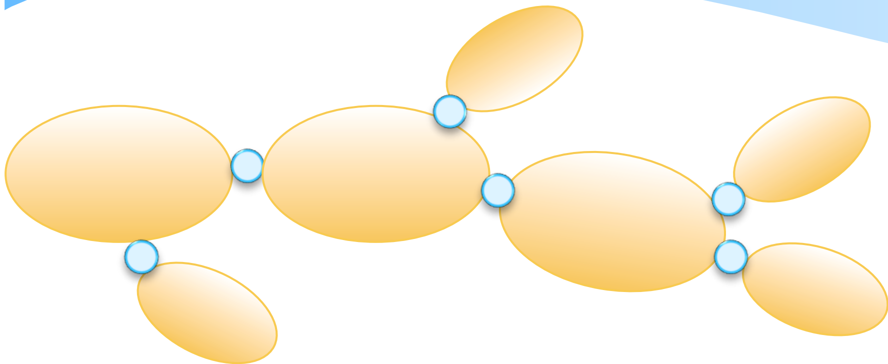
Motivations:

* Cons: classical dynamics computations (e.g. inverse dynamics) rely on a subset of available sensors (e.g. inverse dynamics).
* Pros: classical computations are computationally efficient, e.g. RNEA is O(n).

Problem:

* Efficiently compute the dynamics by exploiting all available sensors, including force/torque sensors, gyros and accelerometers.

Notation



N_B: number of links

Joint quantities

$\boldsymbol{q}_i$: the joint i position,

$\dot{\boldsymbol{q}}_i$: the joint i velocity,

$\ddot{\boldsymbol{q}}_i$: the joint i acceleration,

$\boldsymbol{\tau}_i$: the joint i torque.

Link quantities

$\boldsymbol{v}_i$: the link spatial velocity,

$\boldsymbol{a}_i$: the link spatial accelerations,

$\boldsymbol{f}_i$: the spatial force transmitted to body i from $\lambda(i)$,

$\boldsymbol{f}_i^x$: external forces acting on body i.

Dynamic variables

$$\boldsymbol{d}_i = \begin{bmatrix} \boldsymbol{a}_i \\ \boldsymbol{f}_i \\ \boldsymbol{\tau}_i \\ \boldsymbol{f}_i^x \\ \ddot{\boldsymbol{q}}_i \end{bmatrix}, \quad \boldsymbol{d} = \begin{bmatrix} \boldsymbol{d}_1 \\ \vdots \\ \boldsymbol{d}_{N_B} \end{bmatrix}$$

Joint positions

$$\boldsymbol{q} = \begin{bmatrix} \boldsymbol{q}_1 \\ \vdots \\ \boldsymbol{q}_{N_B} \end{bmatrix}$$

Joint velocities

$$\dot{\boldsymbol{q}} = \begin{bmatrix} \dot{\boldsymbol{q}}_1 \\ \vdots \\ \dot{\boldsymbol{q}}_{N_B} \end{bmatrix}$$

Spatial transformations

Spatial transformations

${}^{j}\boldsymbol{X}_i$: motion-vector transform from link i to j,

${}^{j}\boldsymbol{X}_i^*$: force-vector transform from link i to j,

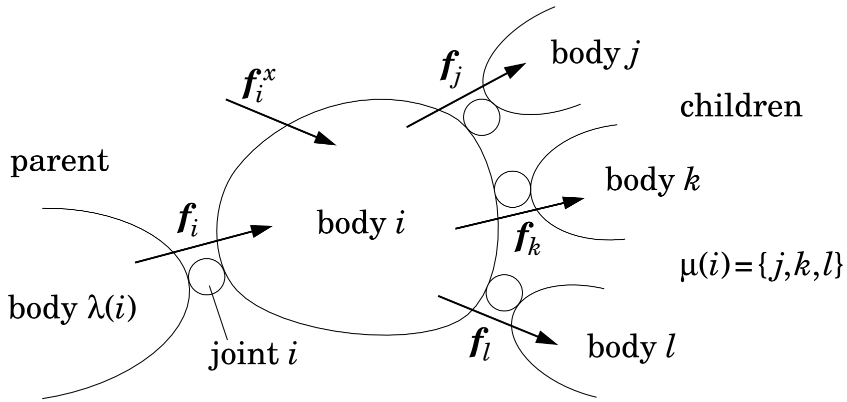
$\boldsymbol{I}_i$: spatial inertia tensor link i,

$(\boldsymbol{v})_l$: linear component,

$(\boldsymbol{v})_a$: angular component,

$\times$: cross product on spatial motions,

$\times^*$: cross product on spatial forces.



Link velocities

$$\boldsymbol{v}_i = {}^{i}\boldsymbol{X}_{\lambda(i)}\boldsymbol{v}_{\lambda(i)} + \boldsymbol{S}_i\dot{\boldsymbol{q}}_i, \quad \Rightarrow \quad \boldsymbol{v}_i(\boldsymbol{q}, \dot{\boldsymbol{q}})$$

SEVENTH FRAMEWORK PROGRAMME

Measurement equation and dynamical consistency

Dynamic and kinematic constraints

$$\begin{aligned}
\boldsymbol{\tau}_i &= \boldsymbol{S}_i^\top \boldsymbol{f}_i, \\
\boldsymbol{a}_i &= {}^i\boldsymbol{X}_{\lambda(i)}(\boldsymbol{q}_i)\boldsymbol{a}_{\lambda(i)} + \boldsymbol{S}_i\ddot{\boldsymbol{q}}_i + \boldsymbol{v}_i \times \boldsymbol{S}_i\dot{\boldsymbol{q}}_i, \\
\boldsymbol{f}_i &= \boldsymbol{I}_i\boldsymbol{a}_i - \boldsymbol{f}_i^x + \sum_{j\in\mu(i)} {}^i\boldsymbol{X}_j^*(\boldsymbol{q}_j)\boldsymbol{f}_j + \boldsymbol{v}_i \times^* \boldsymbol{I}_i\boldsymbol{v}_i
\end{aligned}
\quad \Rightarrow \quad \boxed{\boldsymbol{D}(\boldsymbol{q})\boldsymbol{d} + \boldsymbol{b}_D(\boldsymbol{q},\dot{\boldsymbol{q}}) = \boldsymbol{0}}$$

Measurement equations

$$\begin{aligned}
\boldsymbol{y}_{acc} &= ({}^s\boldsymbol{X}_i\boldsymbol{a}_i)_l + ({}^s\boldsymbol{X}_i\boldsymbol{v}_i)_a \times ({}^s\boldsymbol{X}_i\boldsymbol{v}_i)_l, \\
\boldsymbol{y}_{gyr} &= {}^sR_i(\boldsymbol{v}_i)_a, \\
\boldsymbol{y}_{fts} &= {}^s\boldsymbol{X}_i^*(\boldsymbol{f}_i - \boldsymbol{I}_{im}\boldsymbol{a}_i - \boldsymbol{v}_i \times^* \boldsymbol{I}_{im}\boldsymbol{v}_i), \\
\boldsymbol{y}_{skn} &= {}^s\boldsymbol{X}_i^*\boldsymbol{P}_\perp^s\boldsymbol{f}_i^x.
\end{aligned}
\quad \Rightarrow \quad \boxed{\boldsymbol{Y}(\boldsymbol{q},\dot{\boldsymbol{q}})\boldsymbol{d} + \boldsymbol{b}_Y(\boldsymbol{q},\dot{\boldsymbol{q}}) = \boldsymbol{y}.}$$

$$\begin{bmatrix} \boldsymbol{D}(\boldsymbol{q}) \\ \boldsymbol{Y}(\boldsymbol{q},\dot{\boldsymbol{q}}) \end{bmatrix} \boldsymbol{d} + \begin{bmatrix} \boldsymbol{b}_D(\boldsymbol{q},\dot{\boldsymbol{q}}) \\ \boldsymbol{b}_Y(\boldsymbol{q},\dot{\boldsymbol{q}}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{y} \end{bmatrix}, \quad \begin{bmatrix} \boldsymbol{D}(\boldsymbol{q}) \\ \boldsymbol{Y}(\boldsymbol{q},\dot{\boldsymbol{q}}) \end{bmatrix} \text{ will be assumed full column rank,}$$

Problem 1: estimation

$\boxed{\textsc{Estimation}: \text{estimate } \boldsymbol{d} \text{ given } \boldsymbol{y}.}$

$$p(\boldsymbol{d}, \boldsymbol{y}) = p(\boldsymbol{d}) p(\boldsymbol{y}|\boldsymbol{d})$$

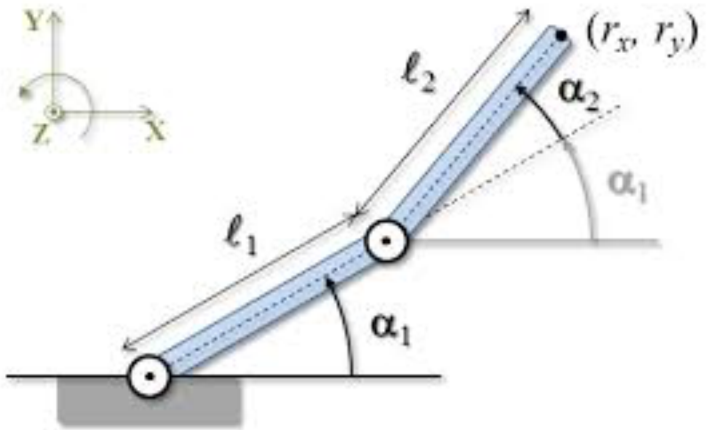
$$\boxed{\boldsymbol{D}(\boldsymbol{q})\boldsymbol{d} + \boldsymbol{b}_D(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{0}} \qquad \boxed{\boldsymbol{Y}(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{d} + \boldsymbol{b}_Y(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{y}.}$$

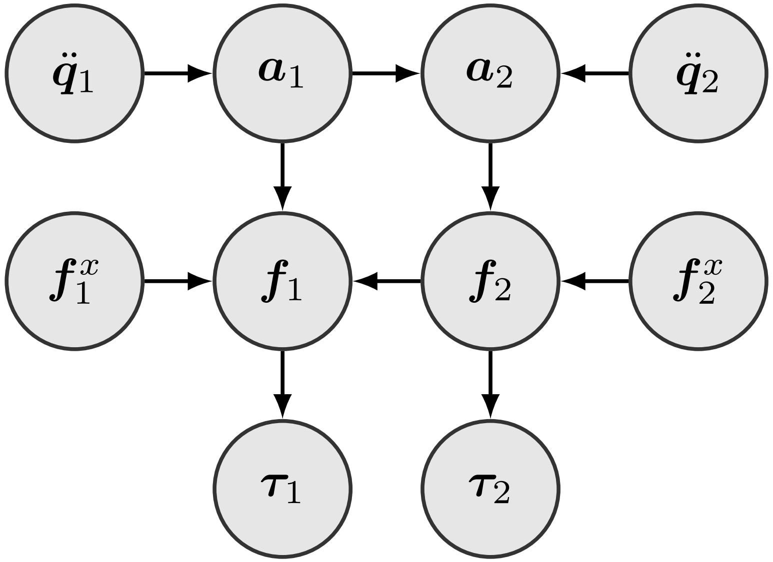
$$\boxed{\begin{aligned} p(\boldsymbol{d}) &\propto \exp\left\{\boldsymbol{e}(\boldsymbol{d})^\top \boldsymbol{\Sigma}_D^{-1} \boldsymbol{e}(\boldsymbol{d})\right\}, \\ \boldsymbol{e}(\boldsymbol{d}) &= \boldsymbol{D}(\boldsymbol{q})\boldsymbol{d} + \boldsymbol{b}_D(\boldsymbol{q}, \dot{\boldsymbol{q}}), \end{aligned}} \qquad \boxed{\begin{aligned} p(\boldsymbol{y}|\boldsymbol{d}) &\sim \mathcal{N}\left(\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y\right), \\ \boldsymbol{\mu}_y &= Y(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{d} + \boldsymbol{b}_Y(\boldsymbol{q}, \dot{\boldsymbol{q}}), \end{aligned}}$$

$$\boldsymbol{d}_{map} = \arg\max_d p(\boldsymbol{d}|\boldsymbol{y}),$$

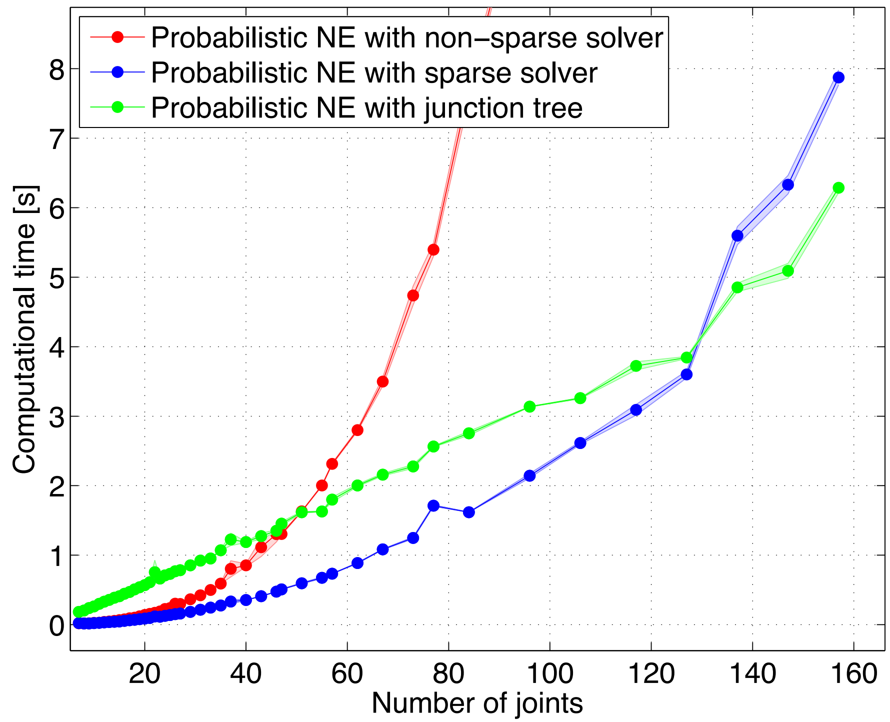
$$\boldsymbol{d}_{map} = \left(\boldsymbol{D}^\top \boldsymbol{\Sigma}_D^{-1} \boldsymbol{D} + \boldsymbol{Y}^\top \boldsymbol{\Sigma}_y^{-1} \boldsymbol{Y}\right)^{-1} \cdot \left(\boldsymbol{Y}^\top \boldsymbol{\Sigma}_y^{-1} (\boldsymbol{y} - \boldsymbol{b}_Y) - \boldsymbol{D}^\top \boldsymbol{\Sigma}_D^{-1} \boldsymbol{b}_D\right).$$

Computational efficiency





Computations have been optimized by exploiting the matrices sparsity.



Open-source code: https://github.com/iron76/bnt_time_varying

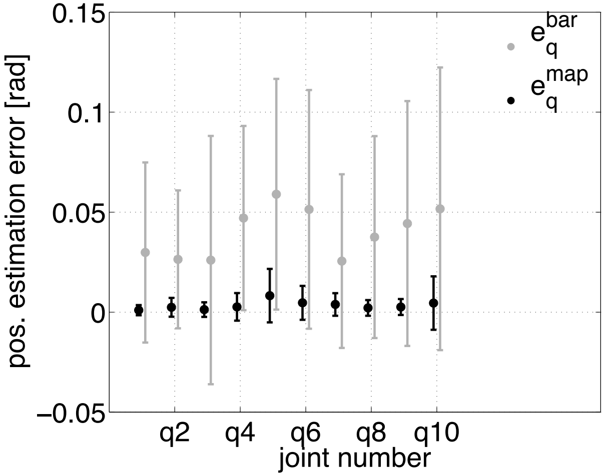
Problem 2: state estimation

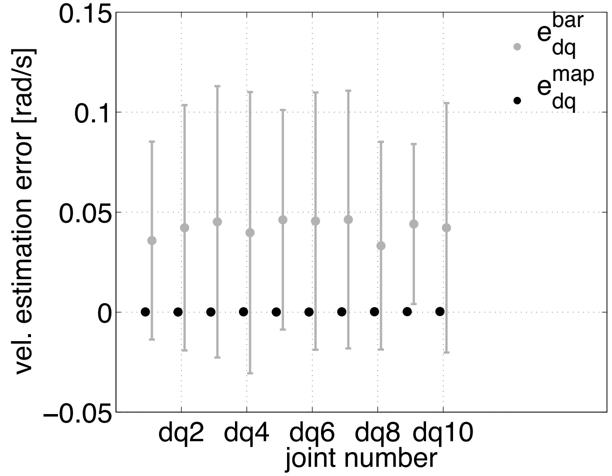
STATE ESTIMATION: estimate $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ given $\boldsymbol{y}$.

$$\begin{bmatrix} \boldsymbol{D}(\boldsymbol{q}) \\ \boldsymbol{Y}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \end{bmatrix} \boldsymbol{d} + \begin{bmatrix} \boldsymbol{b}_D(\boldsymbol{q}, \dot{\boldsymbol{q}}) \\ \boldsymbol{b}_Y(\boldsymbol{q}, \dot{\boldsymbol{q}}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{y} \end{bmatrix}, \qquad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{bmatrix}$$

First order approximation around $\bar{\boldsymbol{d}}$, $\bar{\boldsymbol{x}}$

$$\begin{bmatrix} \boldsymbol{D}(\bar{\boldsymbol{x}}) & \partial \boldsymbol{b}_D(\bar{\boldsymbol{d}}, \bar{\boldsymbol{x}}) \\ \boldsymbol{Y}(\bar{\boldsymbol{x}}) & \partial \boldsymbol{b}_Y(\bar{\boldsymbol{d}}, \bar{\boldsymbol{x}}) \end{bmatrix} \begin{bmatrix} \boldsymbol{d} \\ \boldsymbol{x} \end{bmatrix} + \begin{bmatrix} \boldsymbol{b}_D(\bar{\boldsymbol{x}}) - \partial \boldsymbol{b}_D(\bar{\boldsymbol{d}}, \bar{\boldsymbol{x}})\bar{\boldsymbol{x}} \\ \boldsymbol{b}_Y(\bar{\boldsymbol{x}}) - \partial \boldsymbol{b}_Y(\bar{\boldsymbol{d}}, \bar{\boldsymbol{x}})\bar{\boldsymbol{x}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{y} \end{bmatrix}$$





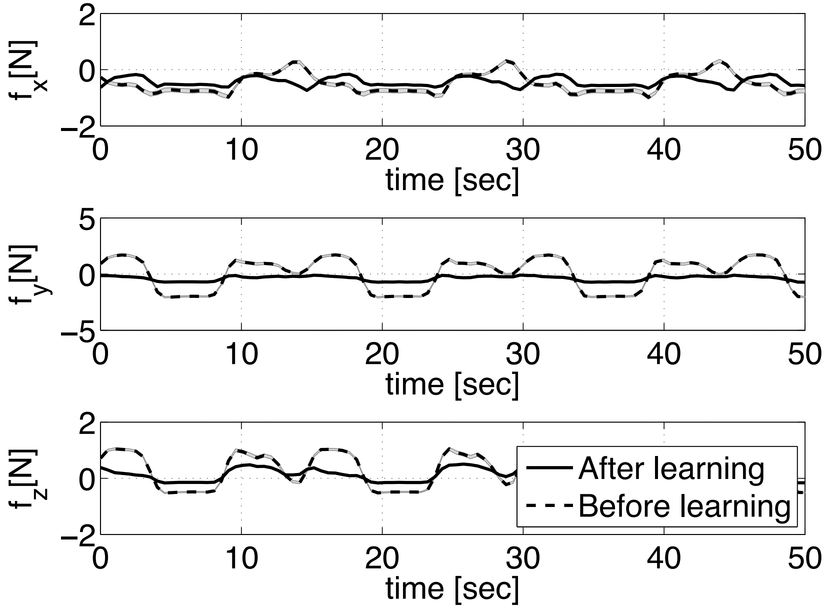
Problem 3: hyper-parameter estimation

HYPERPARAMETER ESTIMATION: estimate $\boldsymbol{\Sigma}_y$ and $\boldsymbol{\Sigma}_d$ given $\boldsymbol{y}$.

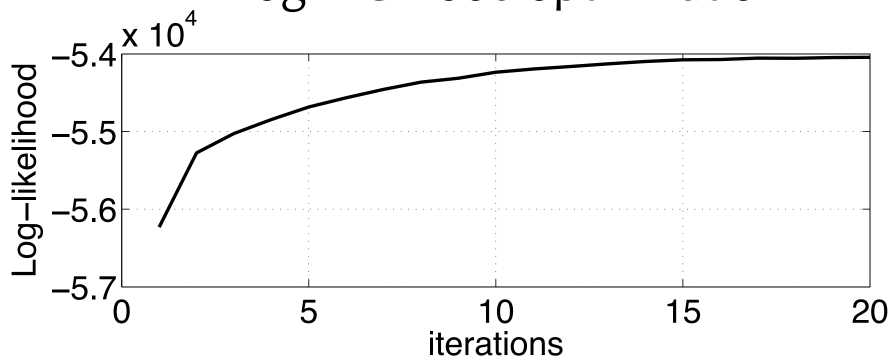
E-step $$Q(\boldsymbol{\theta}|\boldsymbol{\theta}^t) = \sum_t E_{\boldsymbol{d}|\boldsymbol{y}^t;\boldsymbol{\theta}^t}\left[\log p(\boldsymbol{d}, \boldsymbol{y}^t; \boldsymbol{\theta}, t)\right],$$

M-step $$\boldsymbol{\theta}^{t+1} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}|\boldsymbol{\theta}^t).$$

External force estimation



Log-likelihood optimization



Open-source code: https://github.com/iron76/bnt_time_varying