Regression for Robotics

and Motor Adaptation

Olivier Sigaud

ISIR, Sorbonne Université
http://people.isir.upmc.fr/sigaud

January 10, 2022

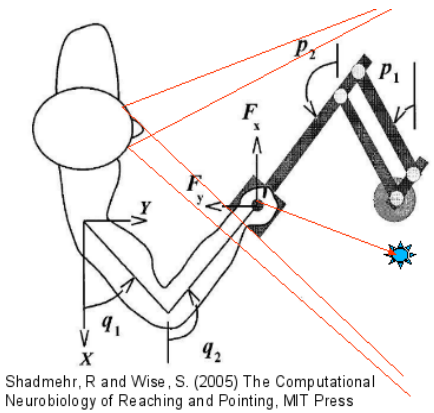
Learning one's body

- Babies don't know well their body

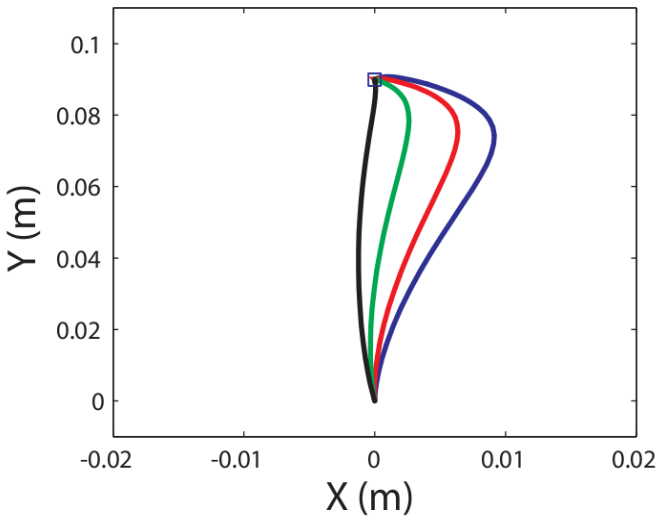
Motor adaptation

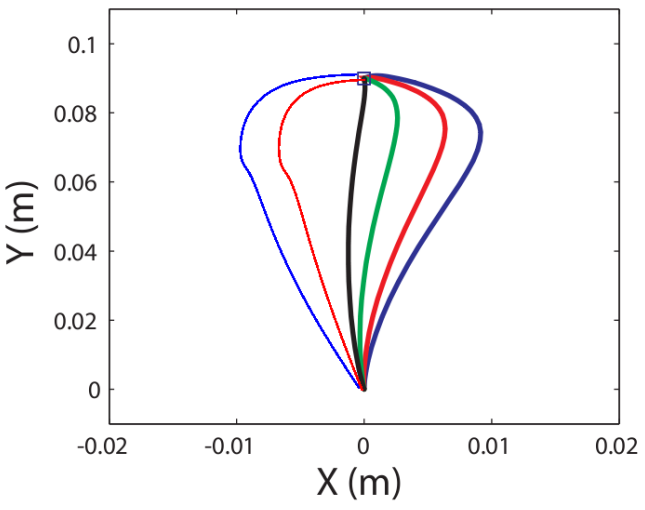
- Adapting one's body model (kinematics, dynamics, ...) under changing circumstances

Motor adaptation: standard experiment



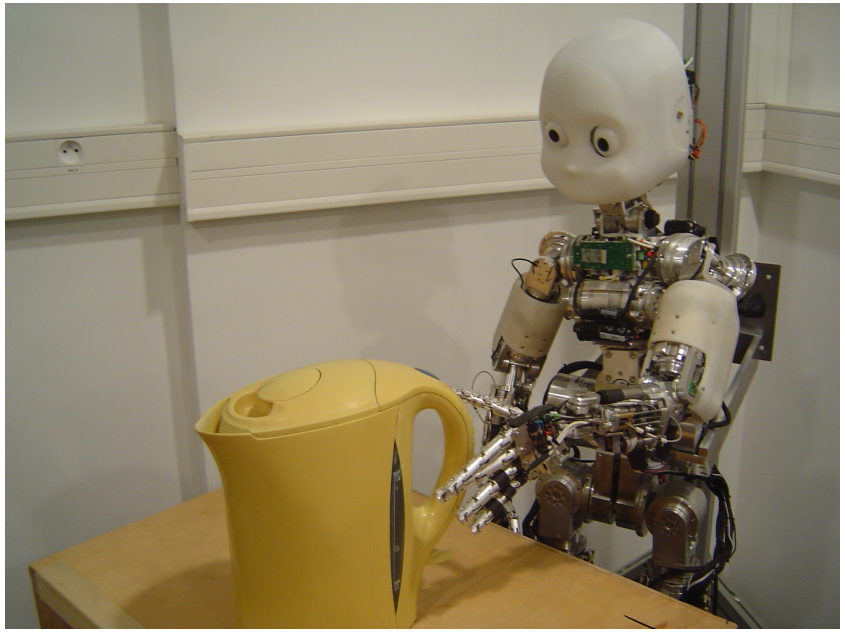
Shadmehr, R and Wise, S. (2005) The Computational Neurobiology of Reaching and Pointing, MIT Press





- Standard view: Motor adaptation results from learning a model of the dynamics

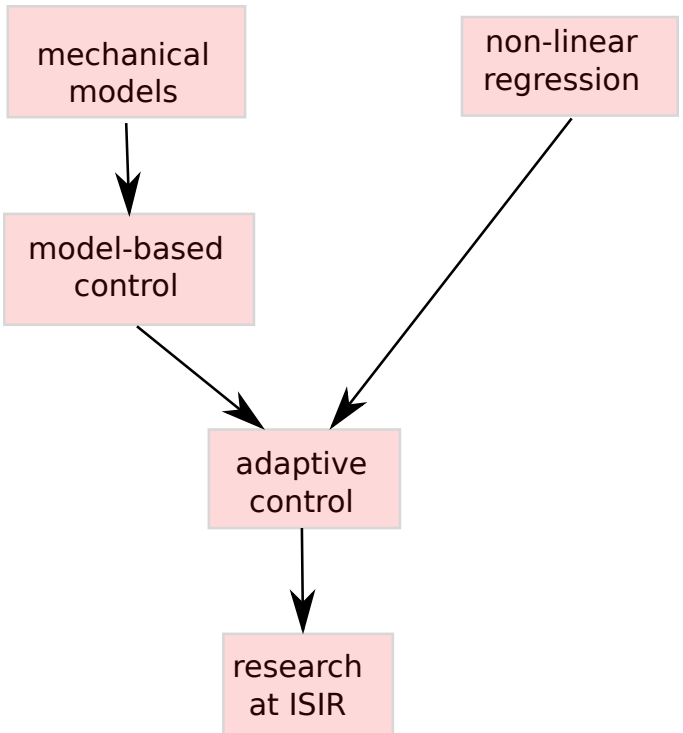
Interest for robotics



Learning interaction models

- Impossible to model unknown objects

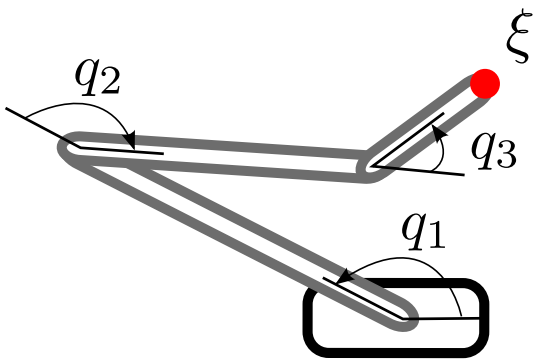
Outline

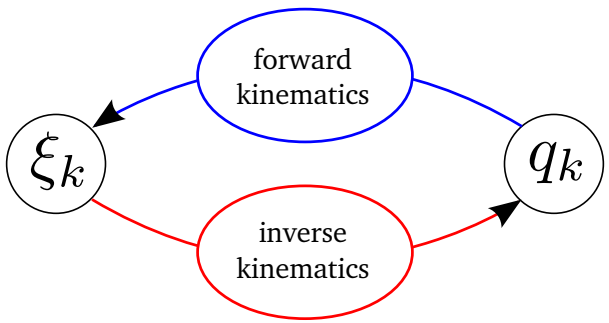


- Tools (regression + control framework) to give a basic account of motor adaptation
- Quick recap on robotics model and control
- Tour of regression algorithms
- Applications

Kinematics

$\boldsymbol{\xi}$: operational position
$\boldsymbol{q}$: articular position

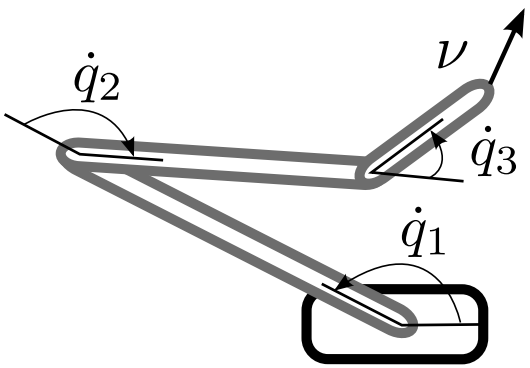


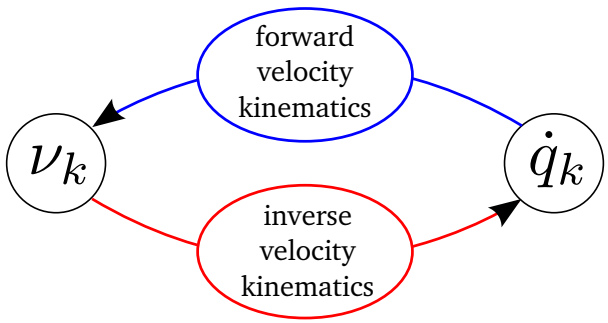


$$\xi_x = l_1 cos(q_1) + l_2 cos(q_1 + q_2) + l_3 cos(q_1 + q_2 + q_3)$$
$$\xi_y = l_1 sin(q_1) + l_2 sin(q_1 + q_2) + l_3 sin(q_1 + q_2 + q_3)$$

Velocity kinematics - Jacobian

$\dot{\boldsymbol{q}}$: articular velocity
$\boldsymbol{\nu}$: operational velocity



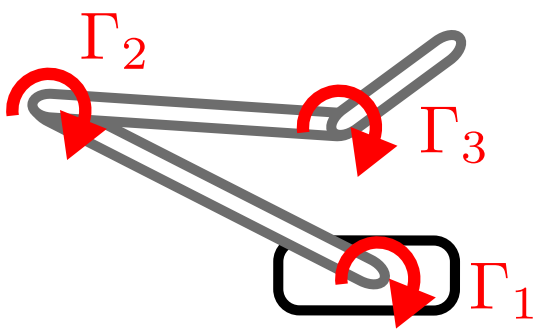


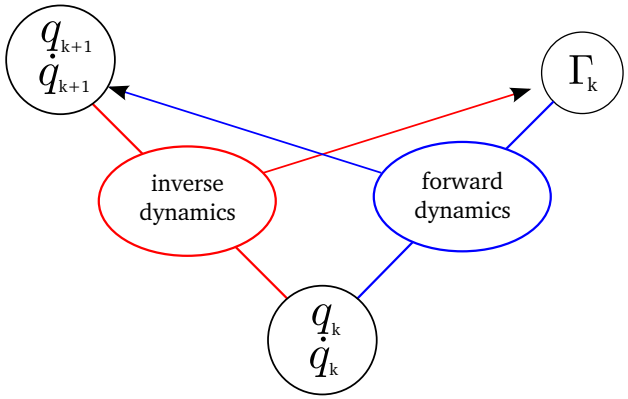
$\nu_x = -(l_1 sin(q_1) + l_2 sin(q_1 + q_2) + l_3 sin(q_1 + q_2 + q_3))\dot{q}_1 - (l_2 sin(q_1 + q_2) + l_3 sin(q_1 + q_2 + q_3))\dot{q}_2 - l_3 sin(q_1 + q_2 + q_3)\dot{q}_3$

$\nu_y = (l_1 cos(q_1) + l_2 cos(q_1 + q_2) + l_3 cos(q_1 + q_2 + q_3))\dot{q}_1 + (l_2 cos(q_1 + q_2) + l_3 cos(q_1 + q_2 + q_3))\dot{q}_2 + l_3 cos(q_1 + q_2 + q_3)\dot{q}_3$

$\boldsymbol{\nu} = J(\boldsymbol{q})\,\dot{\boldsymbol{q}}$

Dynamics: where forces come into play





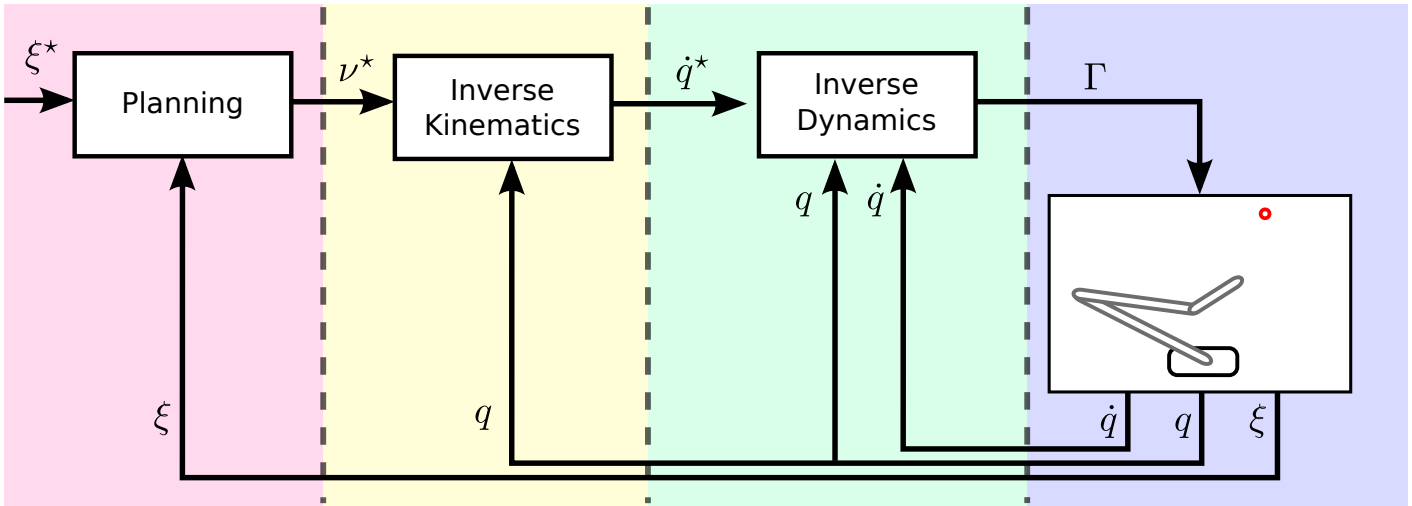
Forward and inverse dynamics (Lagrange or Newton-Euler equations)

$$\ddot{\boldsymbol{q}} = A(\boldsymbol{q})^{-1}\left(\boldsymbol{\tau} - \boldsymbol{n}(\boldsymbol{q},\dot{\boldsymbol{q}}) - \boldsymbol{g}(\boldsymbol{q}) - \boldsymbol{\epsilon}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \boldsymbol{\tau}^{ext}\right)$$
$$\boldsymbol{\tau} = A(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\epsilon}(\boldsymbol{q},\dot{\boldsymbol{q}}) - \boldsymbol{\tau}^{ext}$$

A: inertia matrix
$\boldsymbol{n}$: Coriolis and centrifugal effects
$\boldsymbol{g}$: gravity
$\boldsymbol{\epsilon}$: unmodeled effects
$\boldsymbol{\tau}^{ext}$: external forces

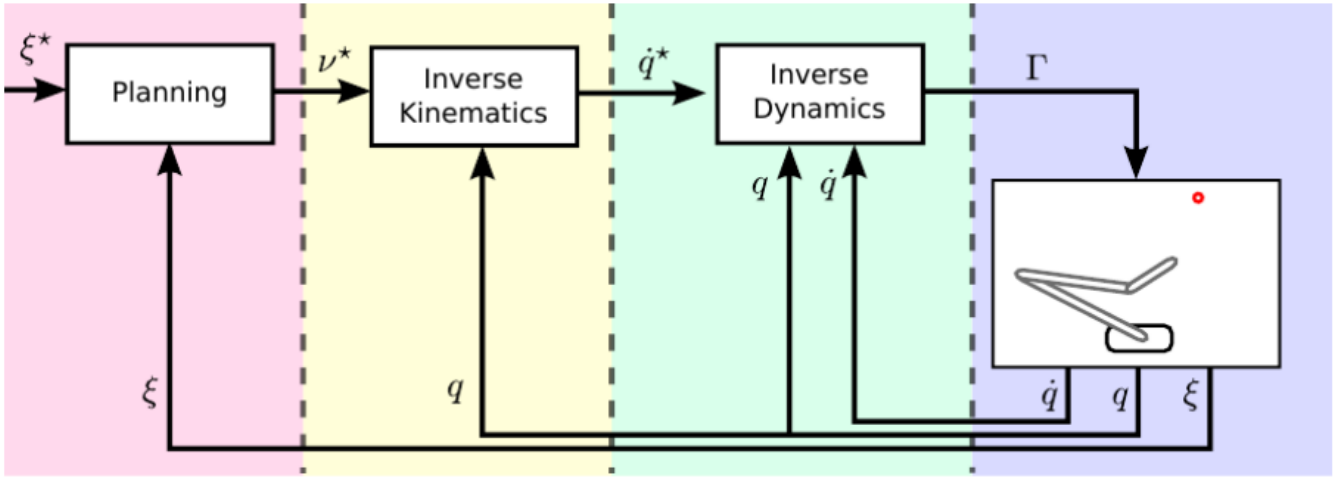
$\boldsymbol{q}$: articular position
$\dot{\boldsymbol{q}}$: articular velocity
$\dot{\boldsymbol{q}}$: articular acceleration
$\boldsymbol{\tau}$: torques

Resolved Motion Rate Control (Whitney 1969)



- Also called CLIK (Closed Loop Inverse Kinematics)
- From task to torques
- Three steps architecture
  - Trajectory generation
  - Inverse Kinematics and redundancy
  - Inverse Dynamics

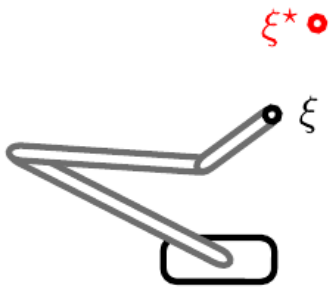
Resolve Motion Rate Control - Trajectory generation

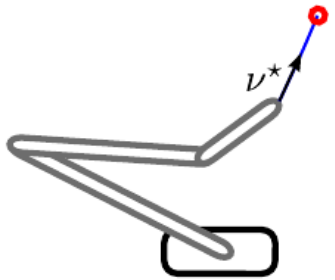


ξ: operational position

$\xi^\dagger$: desired operational position

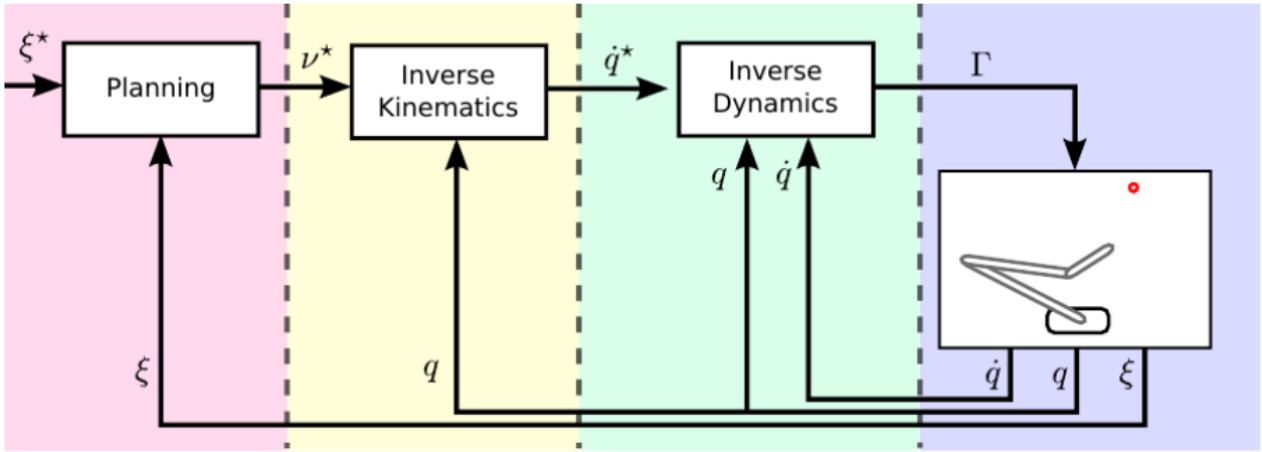
$\nu^\star$: desired operational velocity





First step, create a goal attractor. $\nu^\star = K_p \left(\xi^\dagger - \xi\right)$

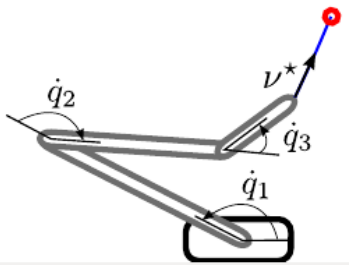
Resolve Motion Rate Control - Inverse kinematics



q: articular position

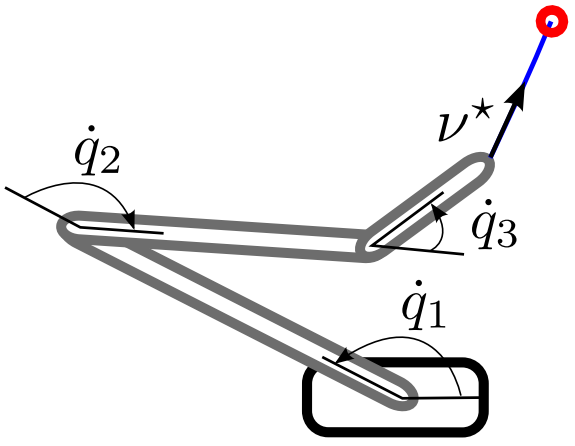
ν: operational velocity

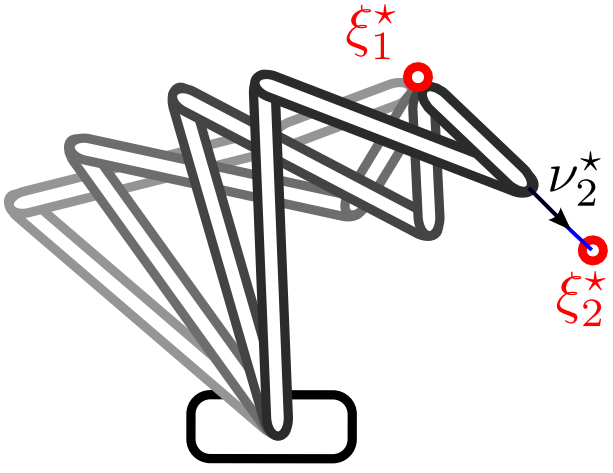
$\dot{q}$: articular velocity



Second step, inverse the kinematics. $\boldsymbol{\nu} = J(\boldsymbol{q})\,\dot{\boldsymbol{q}} \rightarrow \dot{\boldsymbol{q}}^\star = J(\boldsymbol{q})^+\,\boldsymbol{\nu}^\star$

Control redundancy

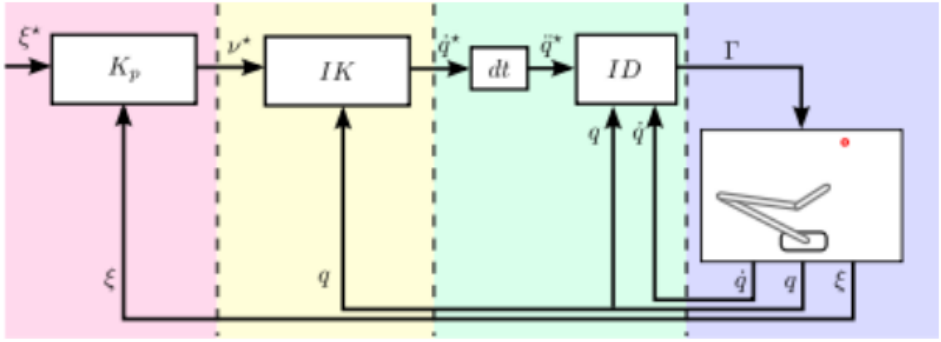




$$\dot{\boldsymbol{q}}^\star = J\left(\boldsymbol{q}\right)^+ \boldsymbol{\nu}^\star \qquad \dot{\boldsymbol{q}}^\star = J_1\left(\boldsymbol{q}\right)^+ \boldsymbol{\nu}_1^\star + \left(J_2\left(\boldsymbol{q}\right) P_{J_1}\right)^+ \boldsymbol{\nu}_2^\star$$

- redundancy : more actuated degrees of freedom than those necessary to realise a task
- P_J is a projector used to control redundancy
- necessary to have access to J to compute P_J

Resolve Motion Rate Control - Inverse Dynamics



Γ: torques

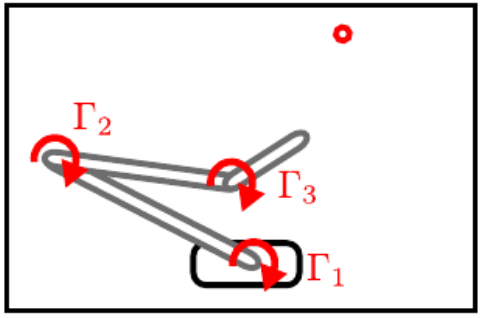
M: inertia matrix

b: Coriolis and centrifugal effects

g: gravity

ϵ: unmodeled effects

Γ^{ext}: external forces



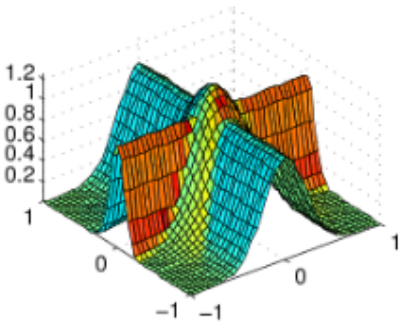
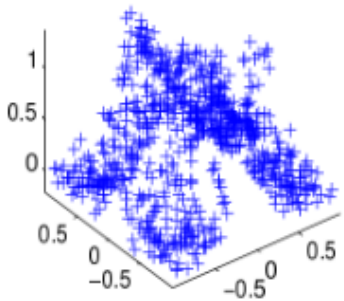
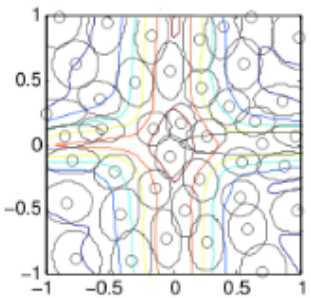
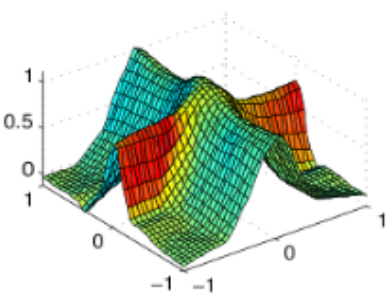
Third step, compute the inverse dynamics

$$\boldsymbol{\tau^{con}} = M(\boldsymbol{q})\,\ddot{\boldsymbol{q}}^\star + \boldsymbol{b}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\epsilon}(\boldsymbol{q},\dot{\boldsymbol{q}}) - \boldsymbol{\tau}^{ext}$$

$$\boldsymbol{\tau^{con}} = \mathcal{ID}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}^\star)$$

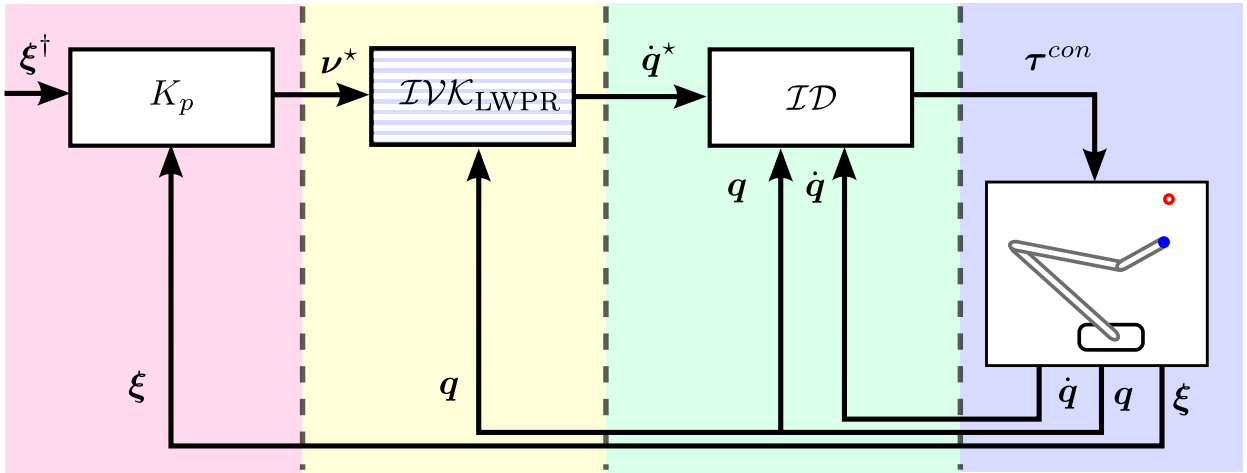
Learning mechanical models

- Forward kinematics: $\dot{\boldsymbol{\xi}} = F_\theta(\boldsymbol{q}, \dot{\boldsymbol{q}})$ $\quad (\dot{\boldsymbol{\xi}} = J(\boldsymbol{q})\,\dot{\boldsymbol{q}})$
- Forward dynamics: $\ddot{\boldsymbol{q}} = G_\theta(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\Gamma})$ $\quad \ddot{\boldsymbol{q}} = A(\boldsymbol{q})^{-1}(\boldsymbol{\Gamma} - \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}))$

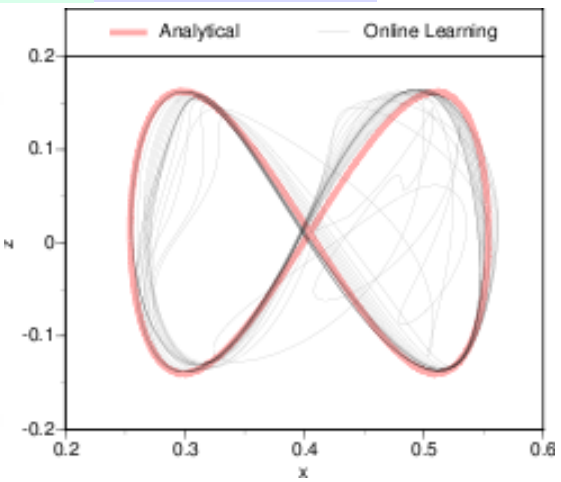





- Regression methods can approximate such functions
- The mapping can be learned incrementally from samples
- Can be used for interaction with unknown objects or users

Learning inverse kinematics with LWPR

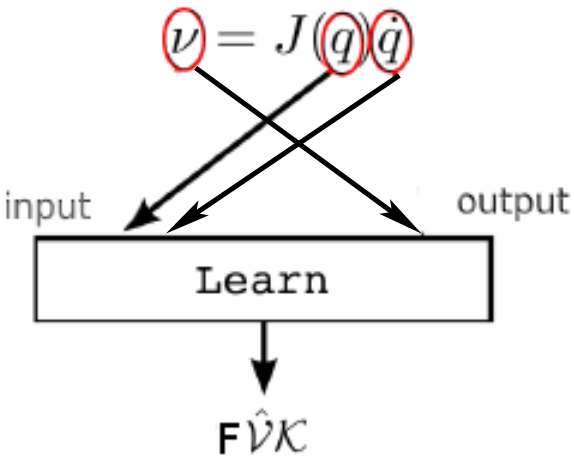


- The model is learned with random movements along an operational trajectory
- Input dimension: $dim(\xi + q) = 29$
- Output dimension: $dim(\dot{q}) = 26$

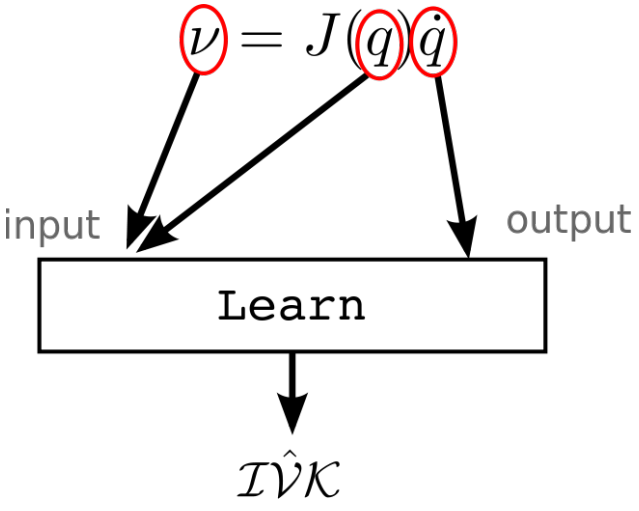


D'Souza, A., Vijayakumar, S., and Schaal, S. (2001b). Learning inverse kinematics. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 298–303.

Learning forward/inverse velocity kinematics with LWPR



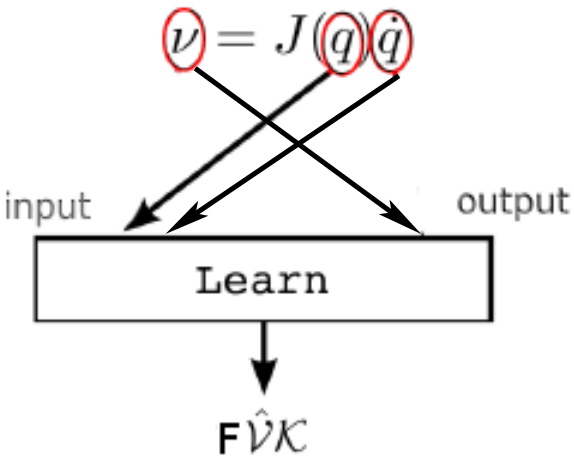
Forward kinematics



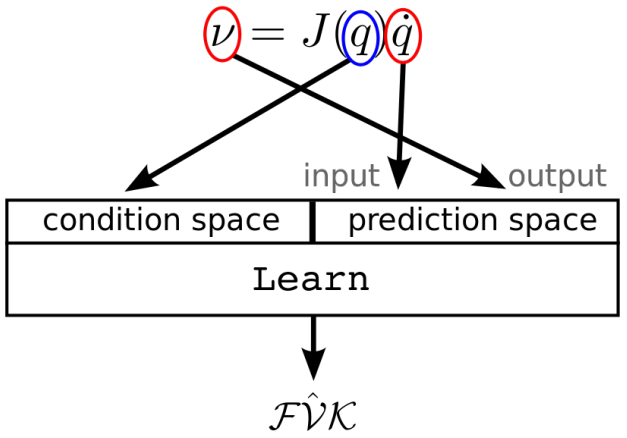
Inverse kinematics

- Learning inverse kinematics is conceptually simpler
- But one loses the opportunity to make profit of redundancy
- Rather learn forward kinematics and inverse it

Learning forward velocity kinematics with LWPR/XCSF



Forward kinematics with LWPR

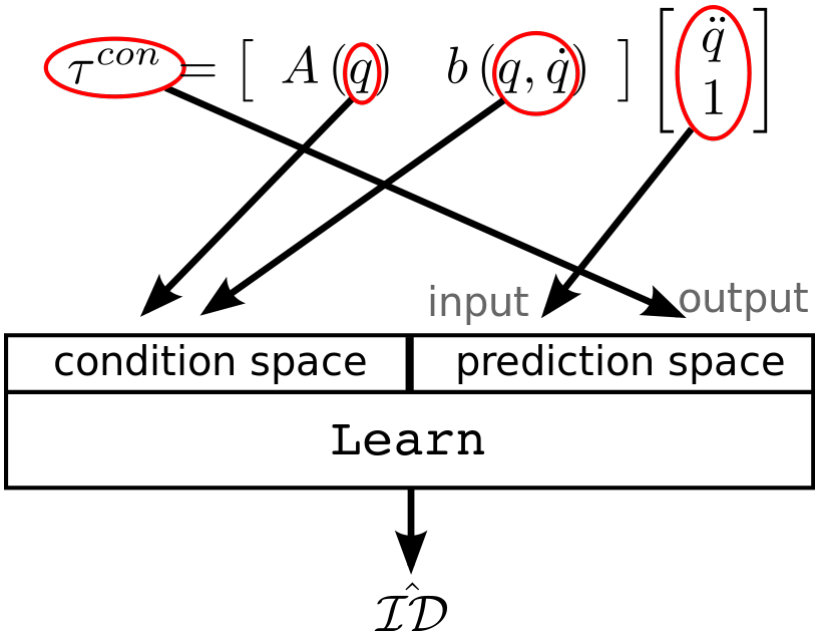


Forward kinematics with XCSF

- Learning the forward velocity kinematics of a Kuka kr16 in simulation.
- They add a constraint to inverse the kinematics and determine the joint velocities.

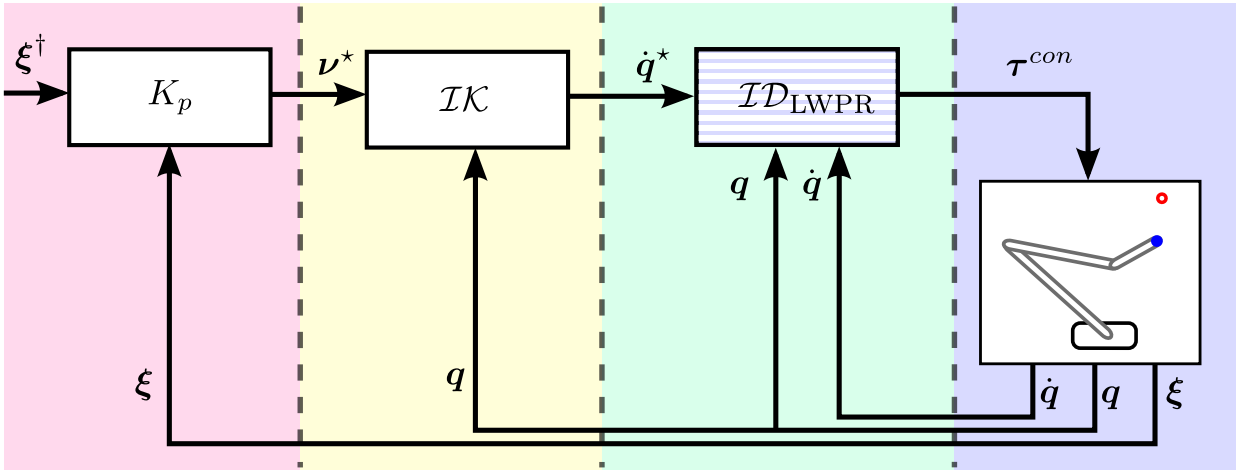
Butz, M., Pedersen, G., and Stalph, P. (2009) Learning sensorimotor control structures with XCSF: redundancy exploitation and dynamic control. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1171–1178. ACM

Learning dynamics with XCSF

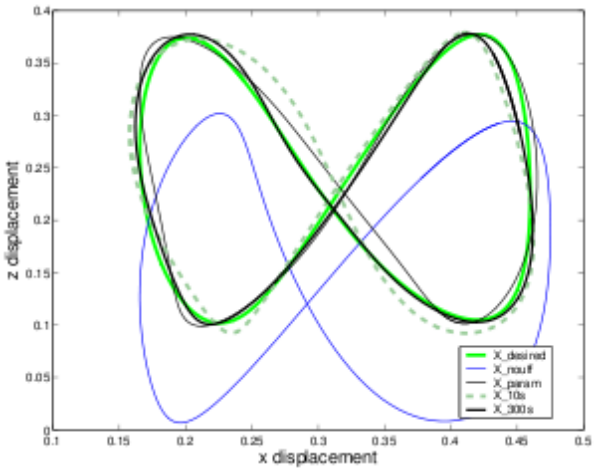


- Learning dynamics is more difficult
- In dynamics, there is no redundancy
- The dynamics model is 2/3 smaller with XCSF than with LWPR

Learning inverse dynamics with LWPR



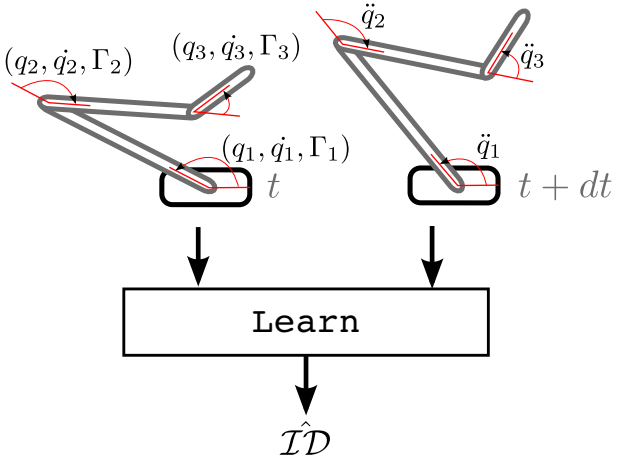
- The model is learned along an operational trajectory
- Input dimension: $dim(q + \dot{q} + \ddot{q}) = 90$
- Output dimension: $dim(\Gamma) = 30$
- $7,5.10^6$ training data points and 2200 receptive fields



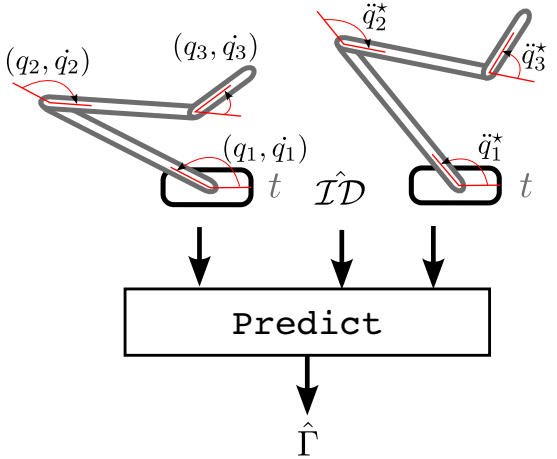
Vijayakumar, S., D'Souza, A., and Schaal, S. (2005). LWPR: A scalable method for incremental online learning in high dimensions. Technical report, Edinburgh: Press of University of Edinburgh.

Learning inverse dynamics

$$\ddot{q} = A(q)^{-1}\left(\tau - n(q,\dot{q}) - g(q) - \epsilon(q,\dot{q}) + \tau^{ext}\right)$$

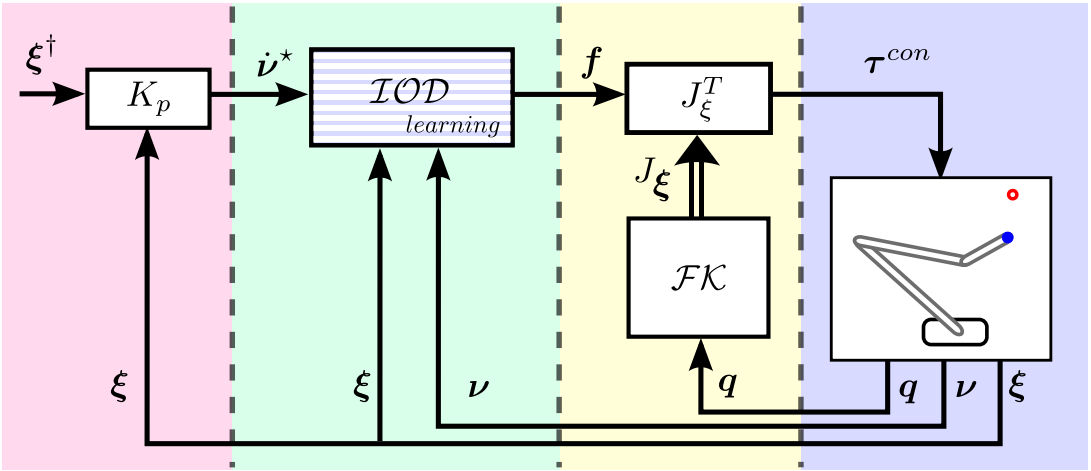


Learning inverse dynamics with random movements along a trajectory

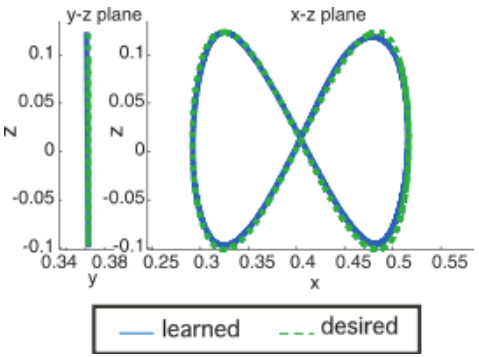


Predict inverse dynamics

Learning inverse operational dynamics

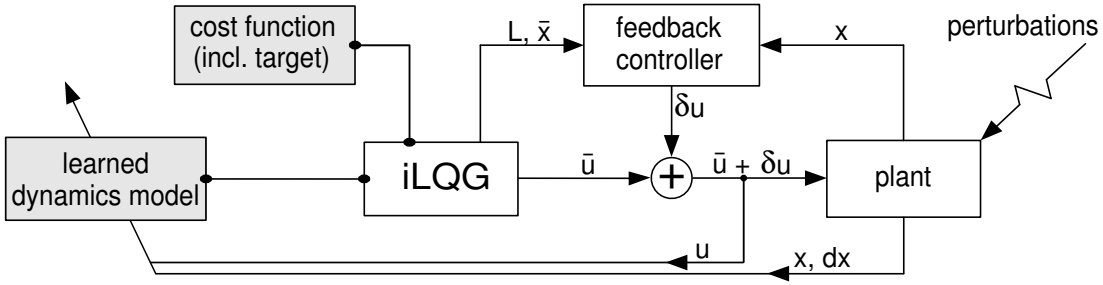


- Peters and Schaal (2008) learn inverse dynamics in the operational space.
- The model is learned along an operational trajectory.
- Input dimension : $dim(q + \dot{q} + \nu) = 17$
- Output dimension: $dim(\Gamma) = 7$

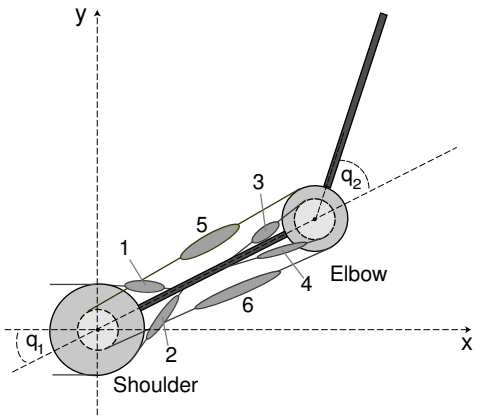


Peters, J. and Schaal, S. (2008). Learning to control in operational space. *International Journal in Robotics Research*, 27(2):197–212.

Optimal control with dynamics learned with LWPR



- The inverse dynamics model is learned in the whole space.
- Input dimension :$dim(q + \dot{q} + u) = 10$. Output dimension : $dim(\ddot{q}) = 2$.
- $1,2.10^6$ training data points and 852 receptive fields
- Learning a model of redundant actuation



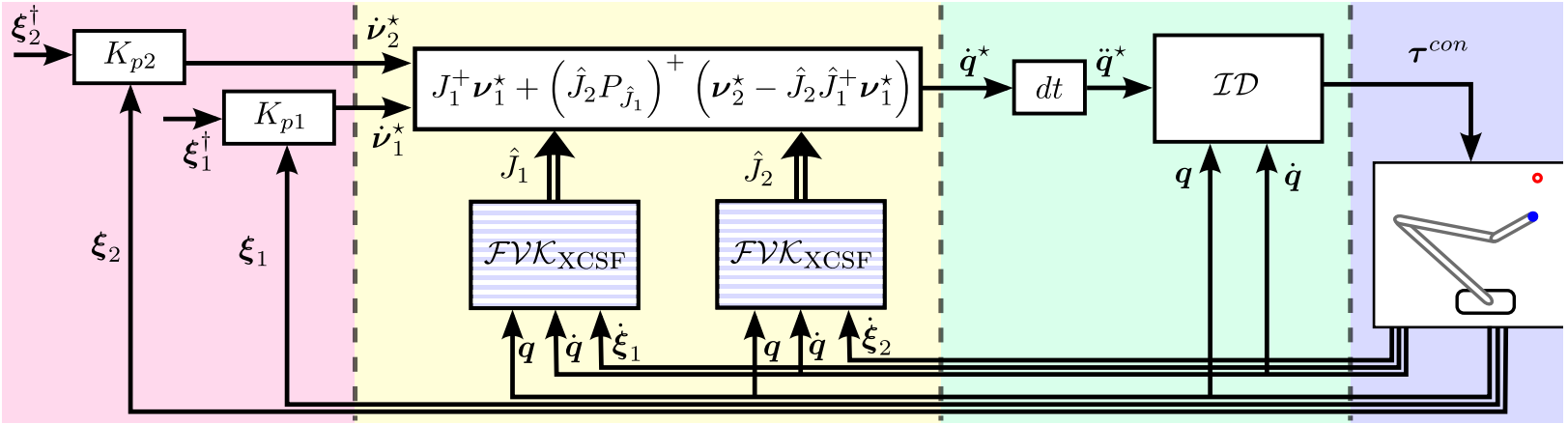
Mitrovic, D., Klanke, S., and Vijayakumar, S. (2008). Adaptive optimal control for redundantly actuated arms. In *Proceedings of the Tenth International Conference on Simulation of Adaptive Behavior*.

Properties of models

	learn forward velocity kinematics	learn inverse velocity kinematics	learn inverse dynamics	learn in whole space	control redundant actuators	control kinematic redundancy
D'Souza et al. (2001)		●				
Vijayakumar et al. (2005)			●			
Peters et al.(2008)		●	●			
Sun et al. (2004)	●			●		
Butz et al. (2009)	●			●		○
Mitrovic et al. (2008)			●		●	
Salaün et al. (2010)	●		●	●		●

- [D'Souza et al., 2001b], [Vijayakumar et al., 2005] and [**?**] learn kinematics and dynamics along a trajectory.
- [Butz et al., 2009] learn kinematics in the whole space but do not make profit of redundancy to combine several tasks.
- [Mitrovic et al., 2008] learn dynamics in the whole space to control redundant actuators.

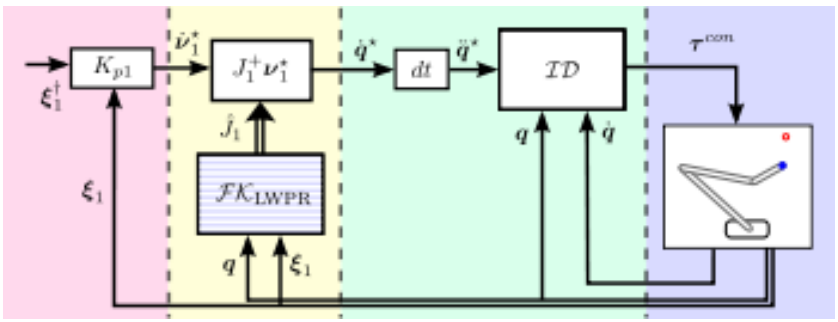
Camille Salaün's work: combining tasks

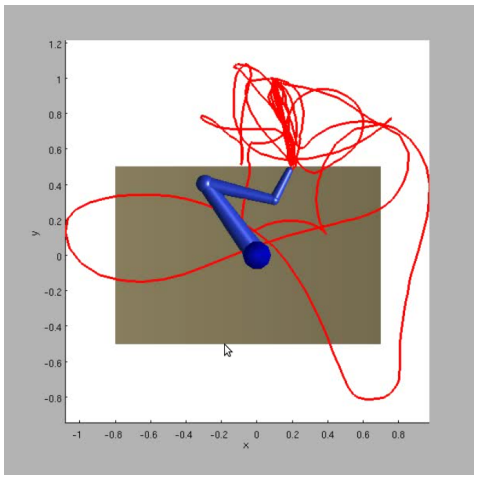


To perform several tasks with learnt models, we have chosen to

- learn separately forward kinematics and inverse dynamics
- use classical mathematical inversion to resolve redundancy
- learn models on whole space
- use LWPR and XCSF as learning algorithms

Learning kinematics with LWPR

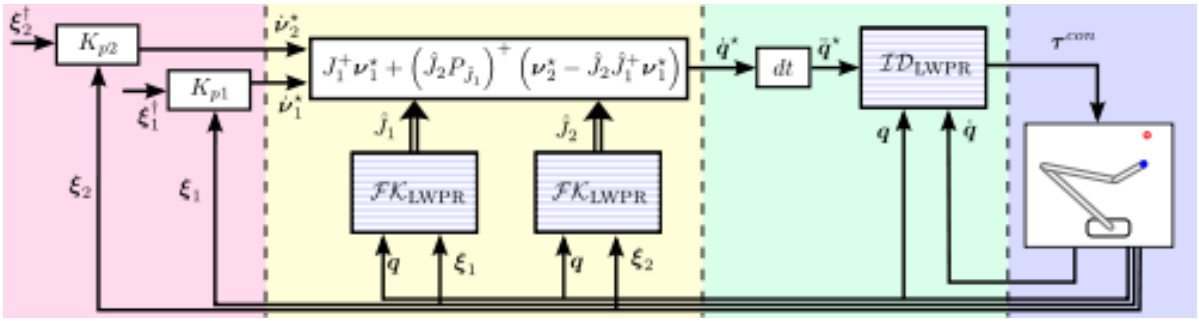


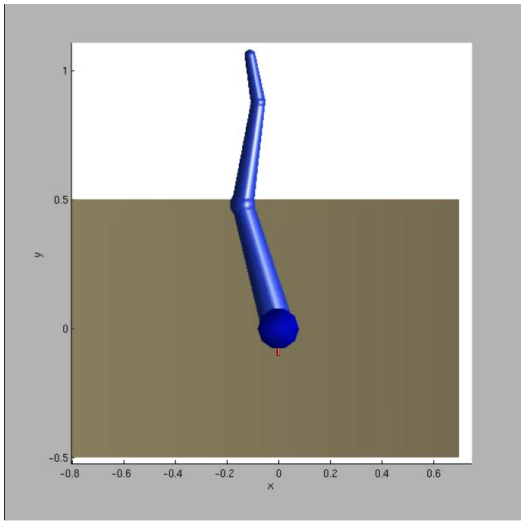


Point to point task

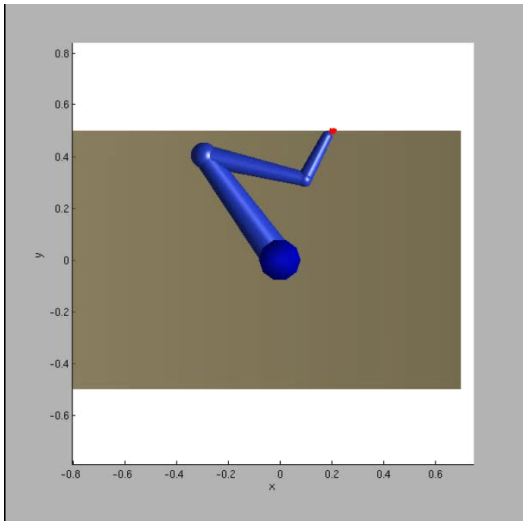
500 steps babbling with the kinematics model we want to learn.

Controlling redundancy with LWPR



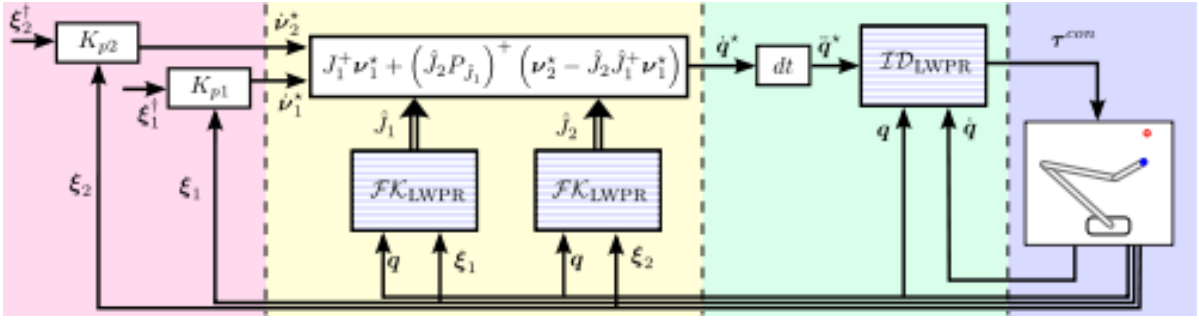


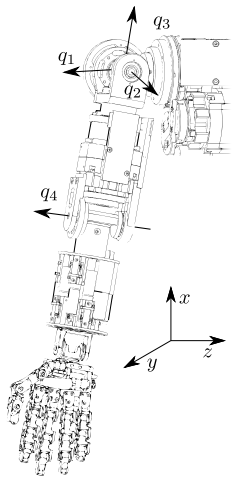
compatible task

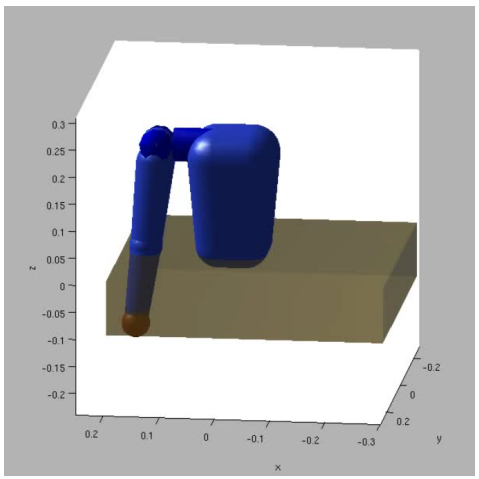


incompatible task

Learning kinematics of iCub in simulation

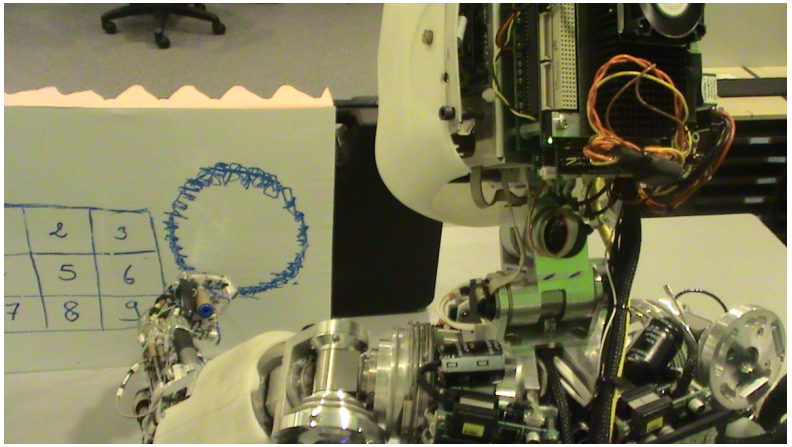


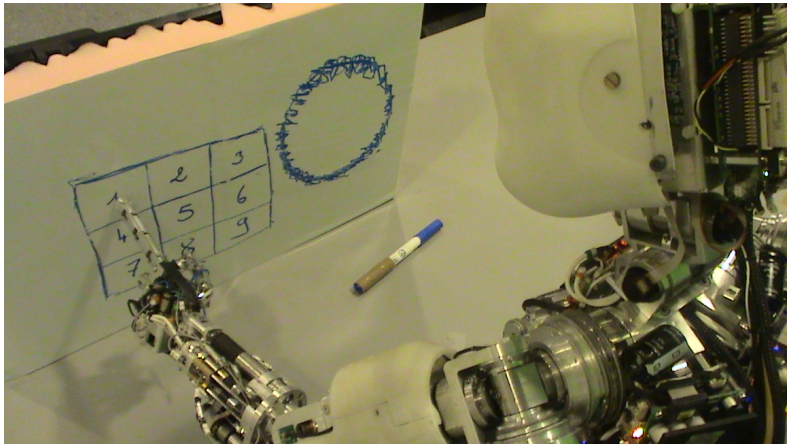


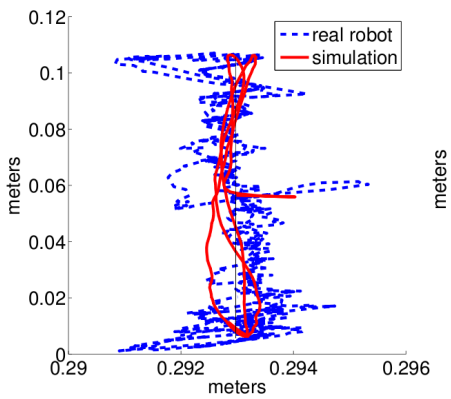


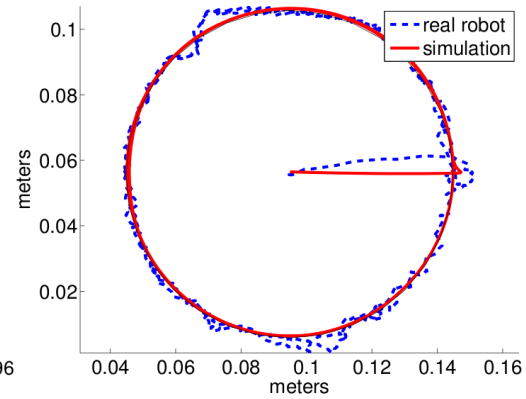
- Simulation of a three degrees of freedom shoulder plus one degrees of freedom elbow

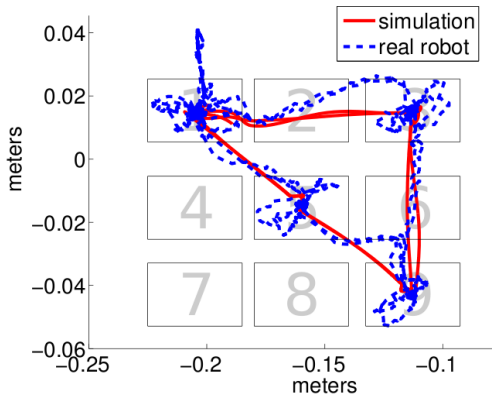
Learning kinematics on the real robot





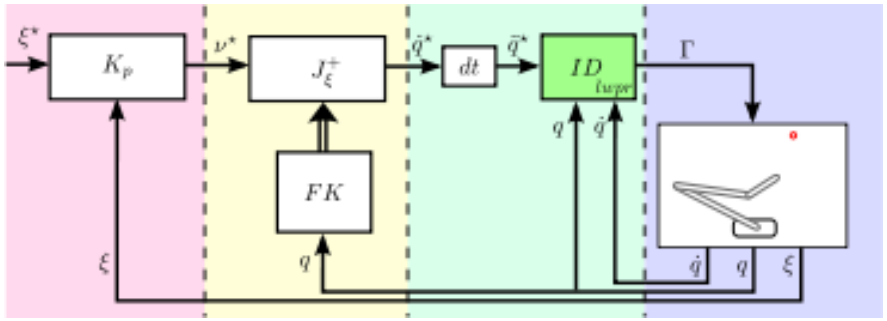


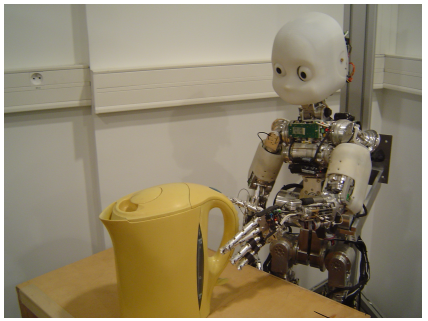


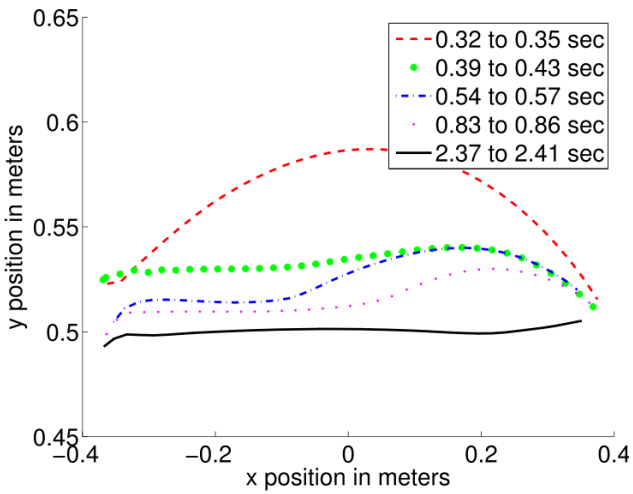


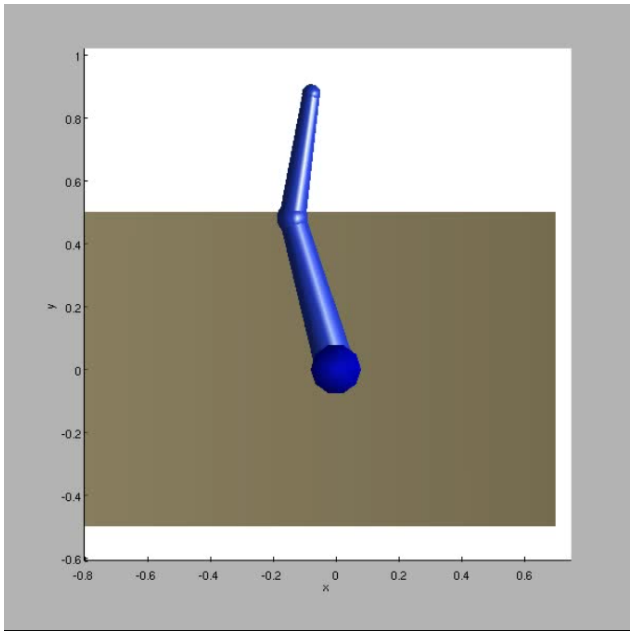
iCub realising two tasks: following a circle and clicking a numpad

Inverse dynamics and motor adaptation



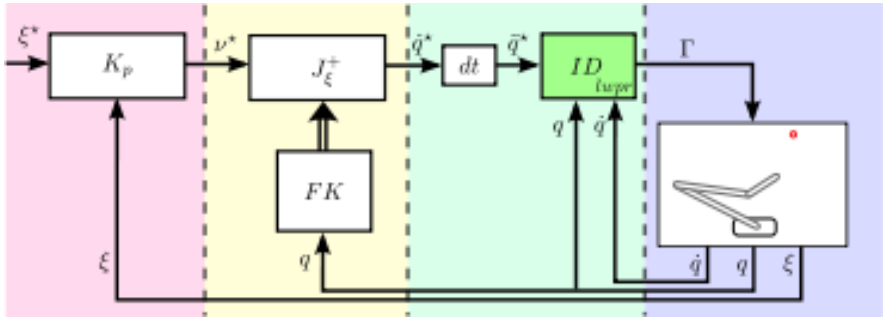


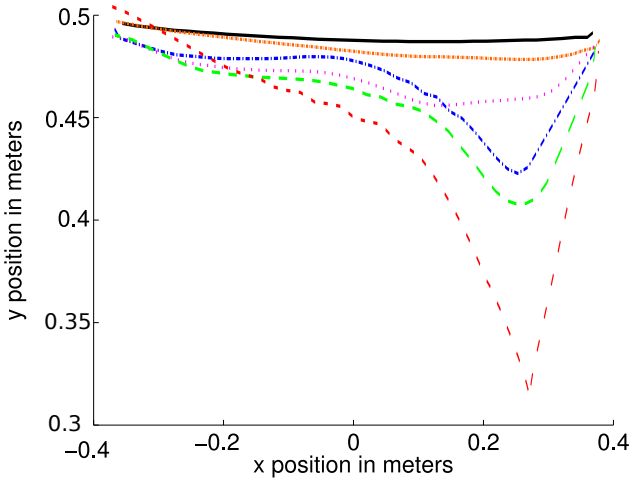


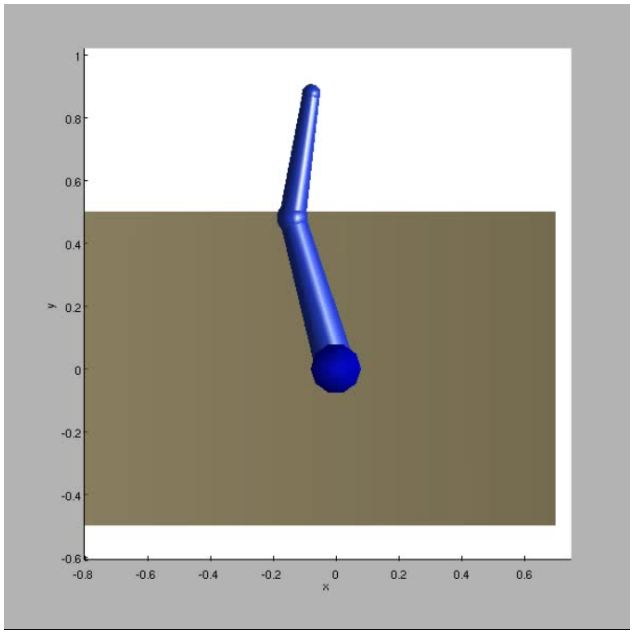


Applying a vertical force after 2 seconds during a point to point task.

Inverse dynamics and after effects



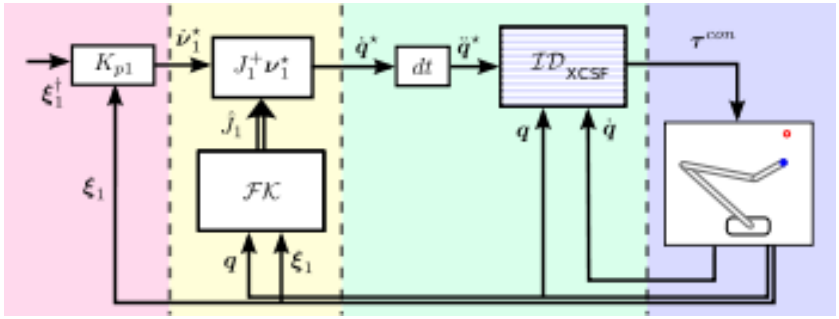




Releasing the force after 2 seconds during a point to point task.

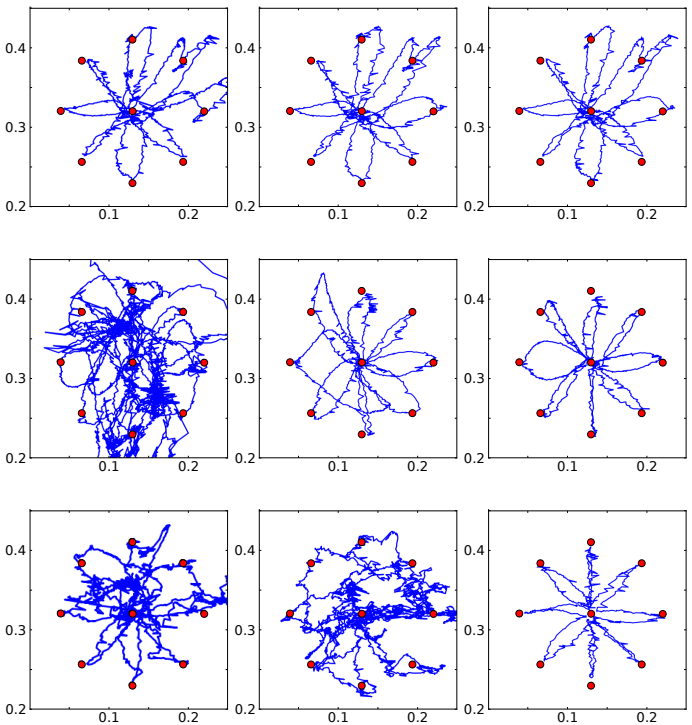
- We reproduce Shadmehr's experiments

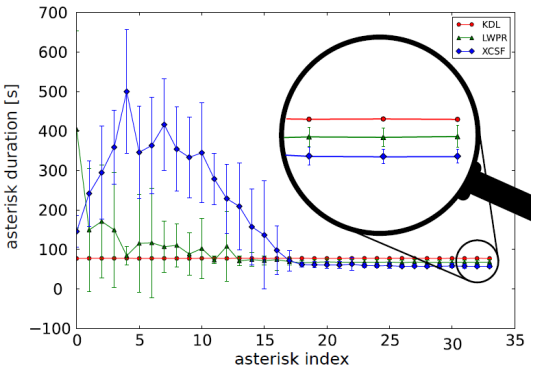
Learning dynamics

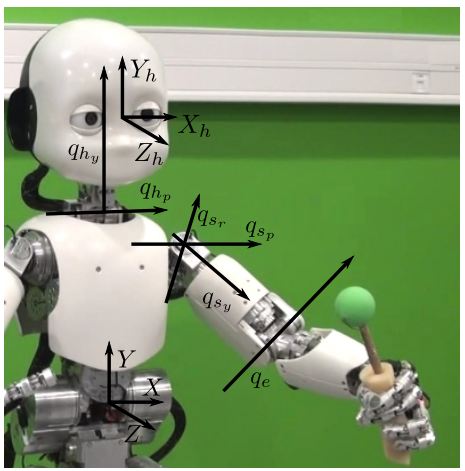


- Simulation of a three degrees of freedom planar arm

Learning forward models



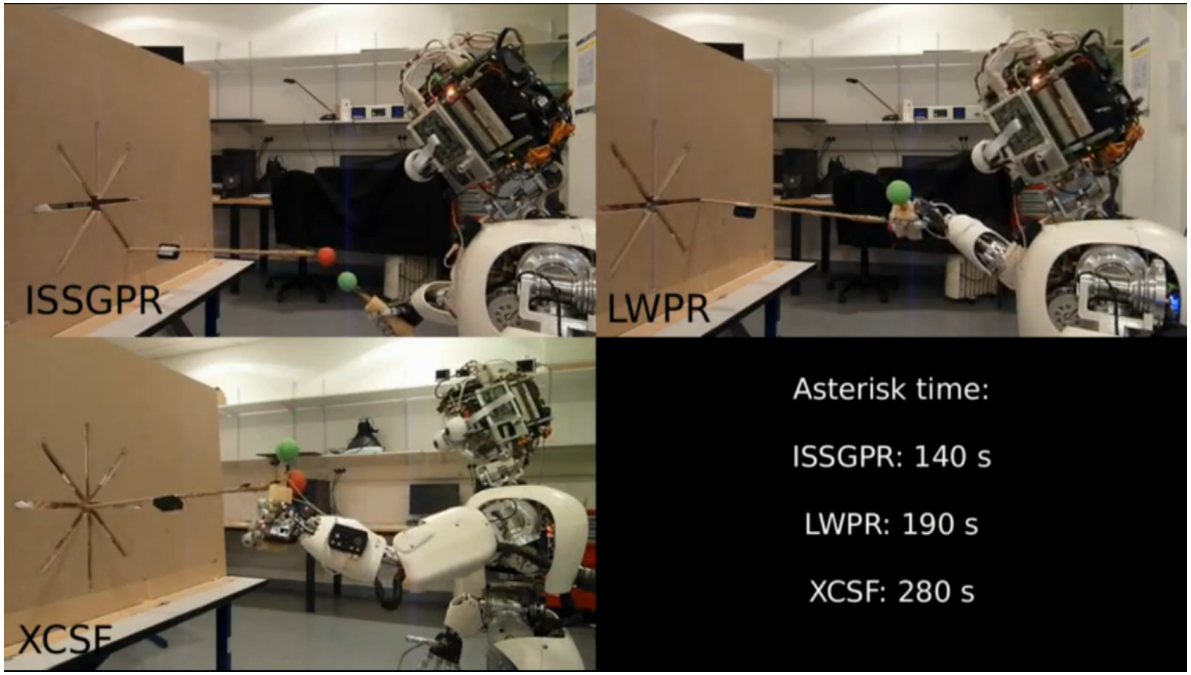




- For complex robots, the CAD model is not so accurate (calibration issue)

Sicard, G., Salaün, C., Ivaldi, S., Padois, V., and Sigaud, O. (2011) Learning the velocity kinematics of icub for model-based control: XCSF versus LWPR. In *Proceedings Humanoids 2011*, pp. 570-575.

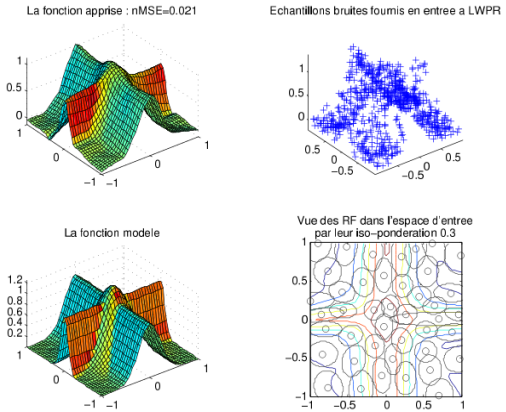
Comparing algorithms

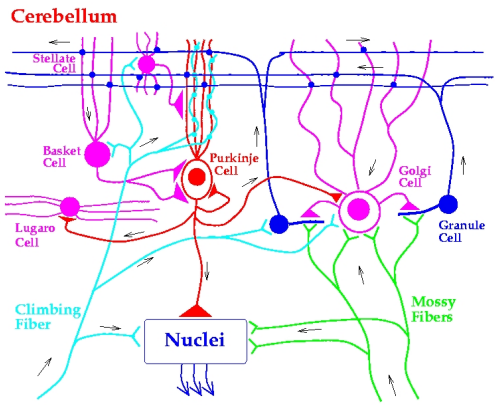


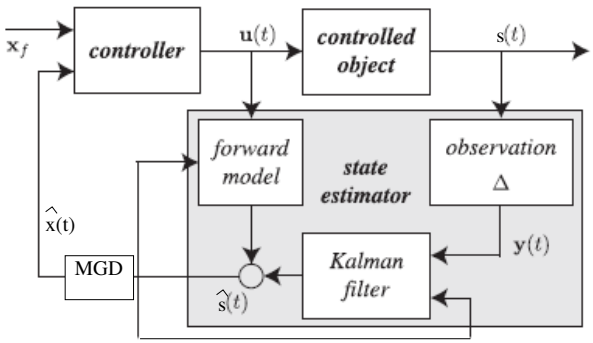
- Main difficulty: tuning parameters for fair comparison
- Many specific difficulties for robotics reproducibility

Droniou, A., Ivaldi, S., Padois, V., and Sigaud, O. (2012) Autonomous Online Learning of Velocity Kinematics on the iCub: a Comparative Study. In *IROS 2012, to appear*

Motor adaptation and the cerebellum

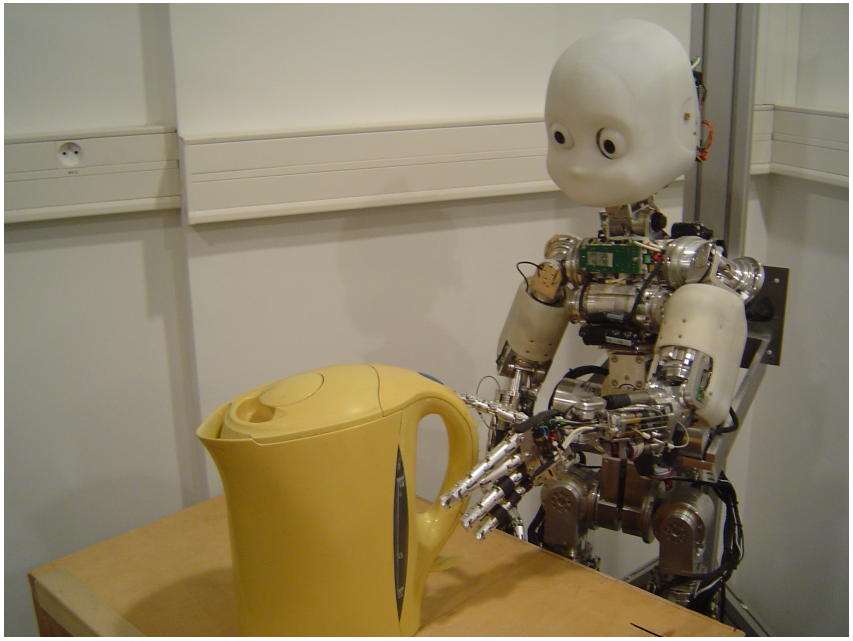






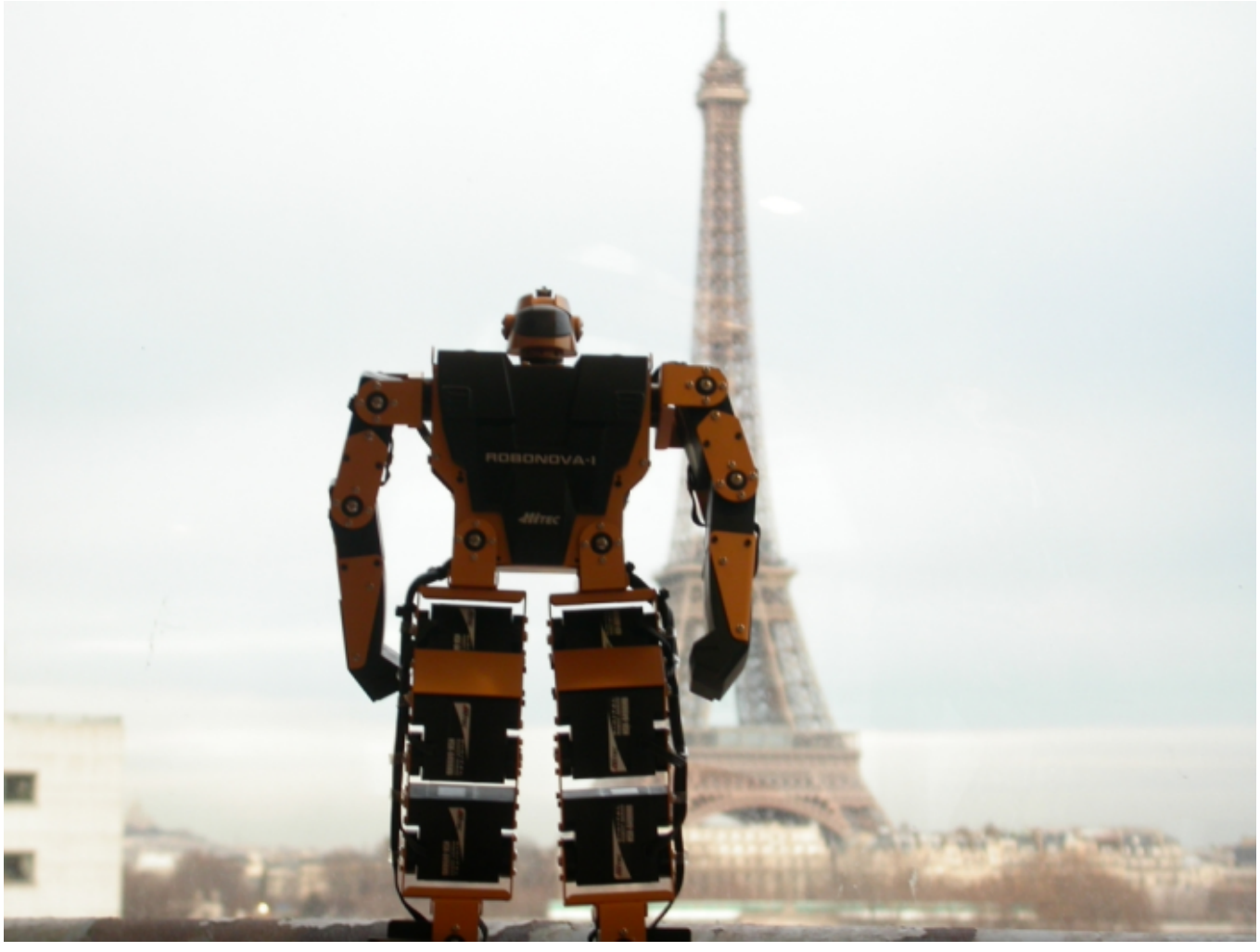
- Structural similarity between LWPR-like algos and cerebellum: Purkinje Cells = receptive fields
- + the problem of state estimation over time given delays

Learning dynamical interactions with objects



- Using a force/torque sensor to detect exerted force on shoulder
- Using artificial skin to detect contact points
- Compliant control of motion (CODYCO EU project)
- Learning high-dimensional models

Any question?



Butz, M., Pedersen, G., & Stalph, P. (2009).

Learning sensorimotor control structures with XCSF: redundancy exploitation and dynamic control.
Edité dans *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1171–1178. ACM.

Droniou, A., Ivaldi, S., Padois, V., & Sigaud, O. (2012).

Autonomous online learning of velocity kinematics on the icub: a comparative study.
Edité dans *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3577–3582, Portugal.

D'Souza, A., Vijayakumar, S., & Schaal, S. (2001a).

Learning inverse kinematics.
Edité dans *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 298–303.

D'Souza, A., Vijayakumar, S., & Schaal, S. (2001b).

Learning inverse kinematics.
Edité dans *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 298–303.

Mitrovic, D., Klanke, S., & Vijayakumar, S. (2008).

Adaptive optimal control for redundantly actuated arms.
Edité dans *Proceedings of the Tenth International Conference on Simulation of Adaptive Behavior*, pages 93–102.

Sicard, G., Salaün, C., Ivaldi, S., Padois, V., & Sigaud, O. (2011).

Learning the velocity kinematics of icub for model-based control: XCSF versus LWPR.
Edité dans *Proceedings of the 11th IEEE-RAS International Conference on Humanoid Robots*, pages 570 – 575, Bled, Slovenia.

Vijayakumar, S., D'Souza, A., & Schaal, S. (2005).

LWPR: A scalable method for incremental online learning in high dimensions.
Rapport technique, Edinburgh: Press of University of Edinburgh.