Motor, Affective and Cognitive Scaffolding for iCub (MACSi)

Project supported by the RobotCub Open Call

Presented at the iCub Workshop

January 18, 2010 / Sestri Levante - Italy

Camille Salaün, Charles Clercq, **Vincent Padois** and Olivier Sigaud

vincent.padois@upmc.fr

Université Pierre et Marie Curie
Institut des Systèmes Intelligents et de Robotique (CNRS UMR 7222)

Origins of the MACSi project

From the industrial to the service Robotics context . . .

Origins of the MACSi project

From the industrial to the service Robotics context . . .

- Programming in advance the behaviour of a robot $\rightarrow$ not viable

Origins of the MACSi project

From the industrial to the service Robotics context . . .

- Programming in advance the behaviour of a robot $\rightarrow$ not viable

$\hookrightarrow$ Endow robots with some **learning capabilities**

Origins of the MACSi project

From the industrial to the service Robotics context . . .

- Programming in advance the behaviour of a robot $\rightarrow$ not viable

$\hookrightarrow$ Endow robots with some **learning capabilities**

- Learning multiple tasks, multiple contexts $\rightarrow \infty$ **of representations**

Origins of the MACSi project

From the industrial to the service Robotics context . . .

- Programming in advance the behaviour of a robot $\rightarrow$ not viable

$\hookrightarrow$ Endow robots with some **learning capabilities**

- Learning multiple tasks, multiple contexts $\rightarrow \infty$ **of representations**
- Designing all such representations by hand $\rightarrow$ impossible

Origins of the MACSi project

From the industrial to the service Robotics context ...

- Programming in advance the behaviour of a robot $\rightarrow$ not viable

$\hookrightarrow$ Endow robots with some **learning capabilities**

- Learning multiple tasks, multiple contexts $\rightarrow \infty$ **of representations**
- Designing all such representations by hand $\rightarrow$ impossible

Need for learning mechanisms able to build increasingly complex problem-specific representations as well as their solution from initially unstructured sensori-motor experiences.

Main goals of the MACSi Project

Central target of MACSi

Build mechanisms allowing a robot:

- to efficiently develop new basic sensorimotor skills
- in partially unknown environments

through both:

- autonomous exploration
- social interaction with humans.

Main goals of the MACSi Project

Central target of MACSi

Build mechanisms allowing a robot:

- to efficiently develop new basic sensorimotor skills
- in partially unknown environments

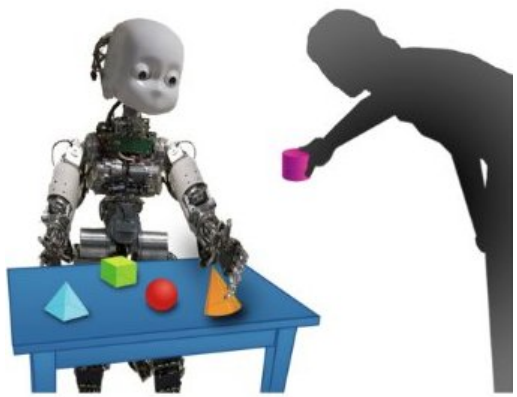
through both:

- autonomous exploration
- social interaction with humans.

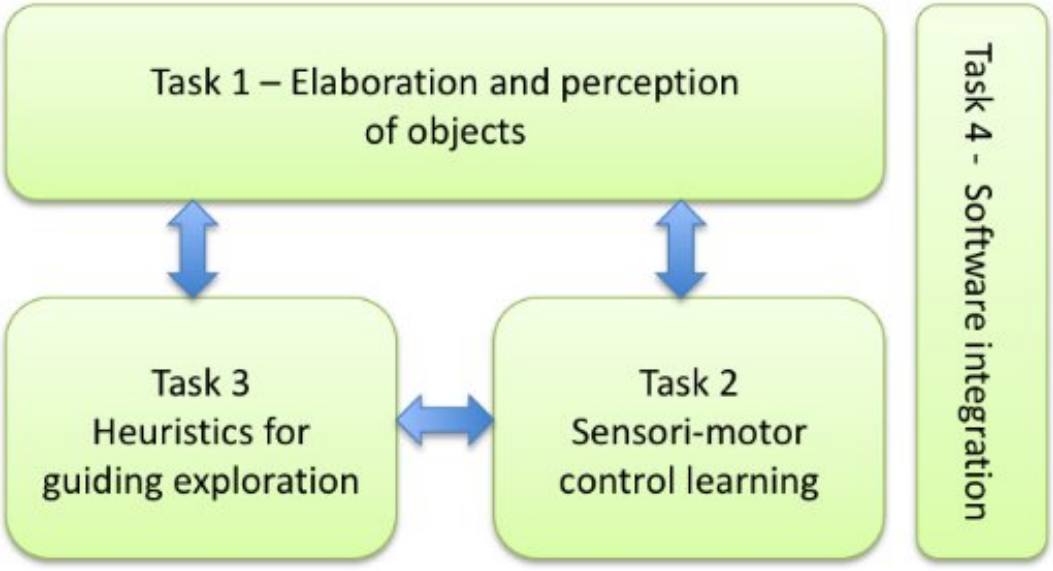
Goal experiment

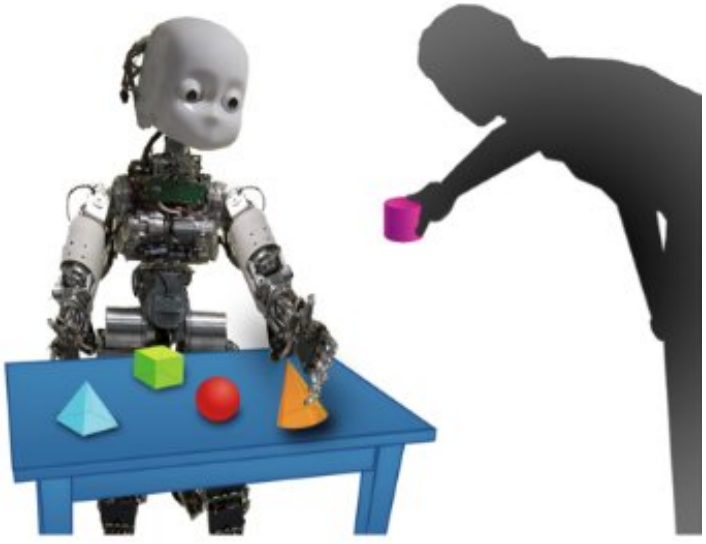
An experiment in which iCub will:

- progressively build perceptuo-motor abstractions and representations allowing iCub to differentiate its body from external objects;
- learn how to control its body to manipulate these surrounding objects;
- driven both by intrinsic motivation, i.e. artificial curiosity, and social guidance provided by a human partner.



Overview of the MACSi project





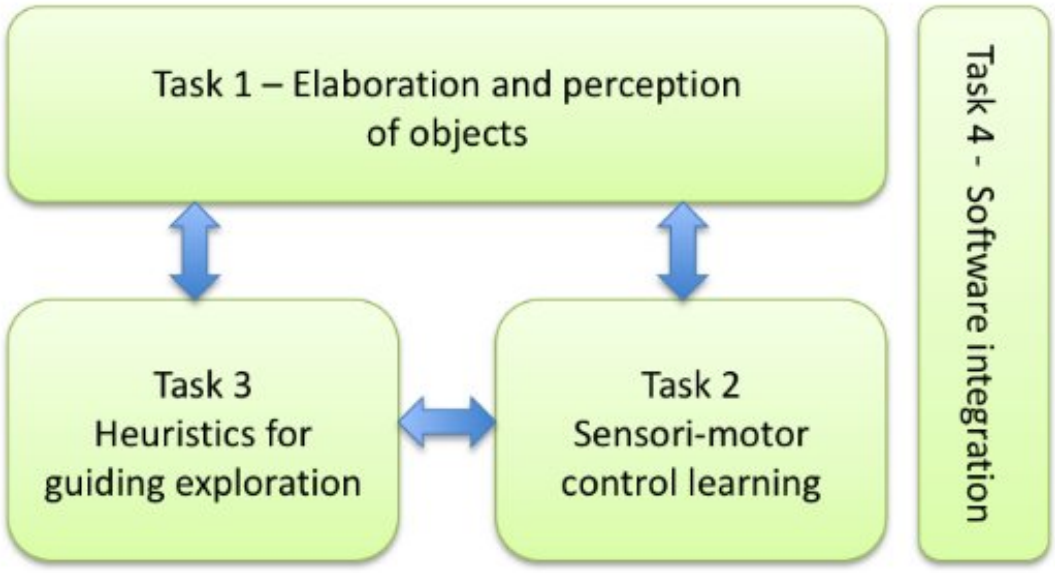
ISIR INSTITUT DES SYSTÈMES INTELLIGENTS ET DE ROBOTIQUE

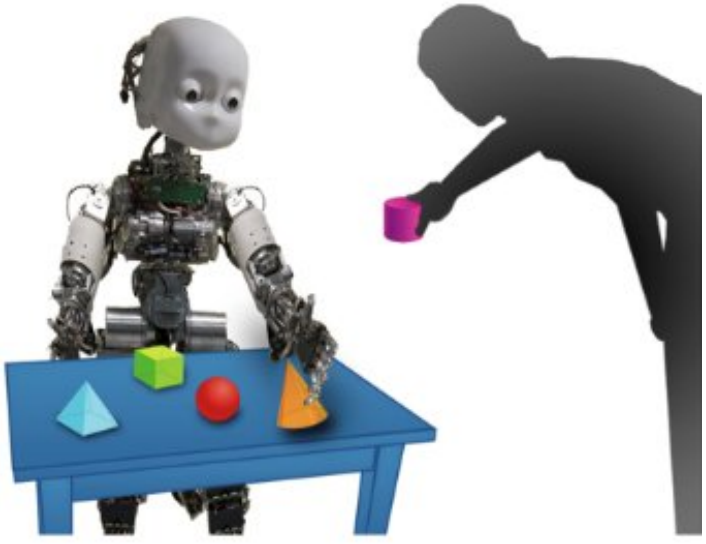
INRIA Flowers

Cognitive Robotics ENSTA

GOSTAI Robotics for everyone

Overview of the MACSi project





ISIR INSTITUT DES SYSTÈMES INTELLIGENTS ET DE ROBOTIQUE

INRIA Flowers

Cognitive Robotics ENSTA

GOSTAI Robotics for everyone

→ In this presentation, focus is put on some of the work related to Task 2.

Models

Models

Which ones ?

- Of the robot itself;
- Of its environment.

Models

Which ones ?

- Of the robot itself;
- Of its environment.

At which level ?

- Robot:
  - 3D geometry;
  - actuators dynamics;
  - joint space to task space direct and inverse mappings: kinematics, velocity kinematics, dynamics;
- Environment:
  - 3D geometry;
  - Interaction dynamics;
  - Behavioural.

Models

Which ones ?

- Of the robot itself;
- Of its environment.

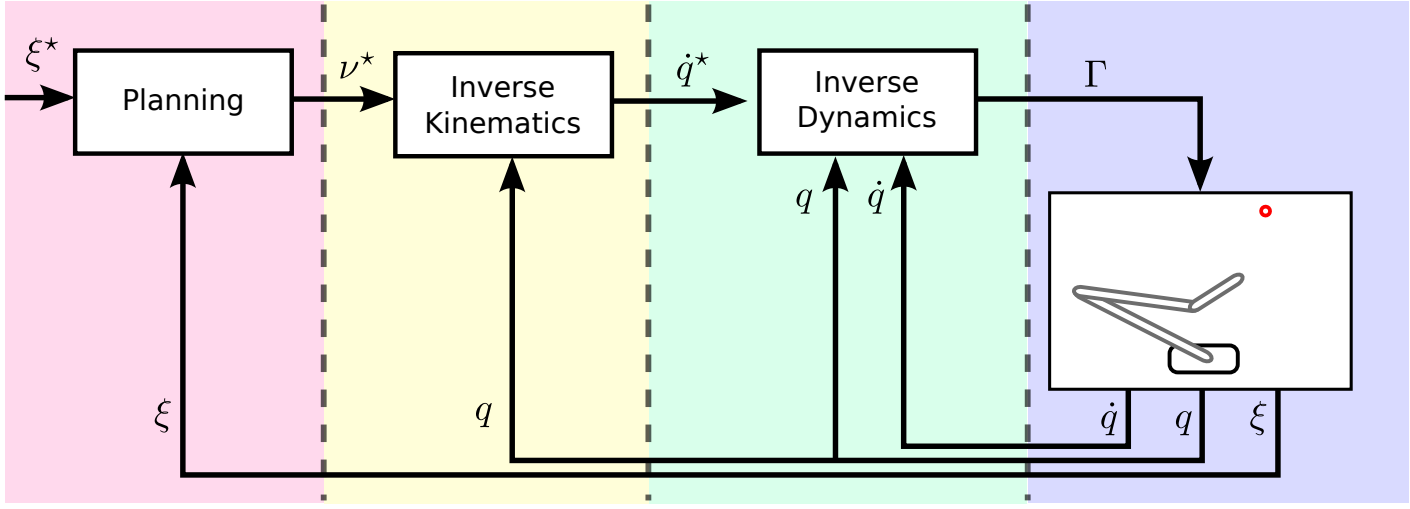
At which level ?

- Robot:
  - 3D geometry;
  - actuators dynamics;
  - joint space to task space direct and inverse mappings: kinematics, velocity kinematics, dynamics;
- Environment:
  - 3D geometry;
  - Interaction dynamics;
  - Behavioural.

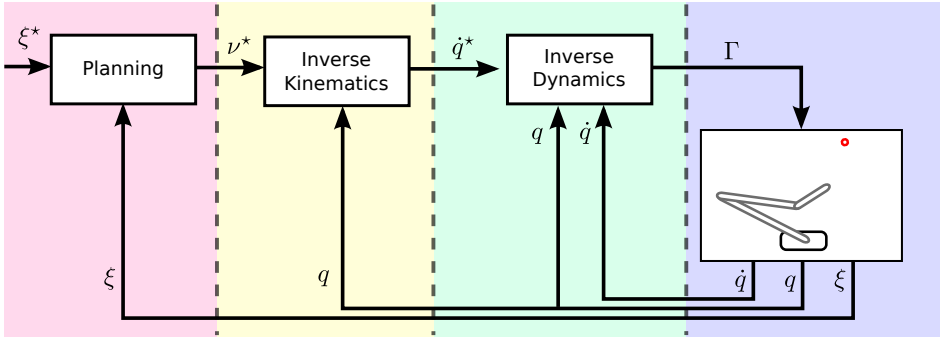
What for ?

- Tasks realization (control);
- Constraints satisfactions;
- Future states prediction.

Control: the velocity level task space loop closing example



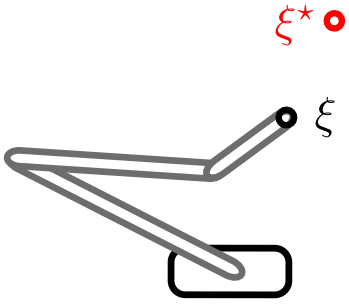
Planning: simplest implementation

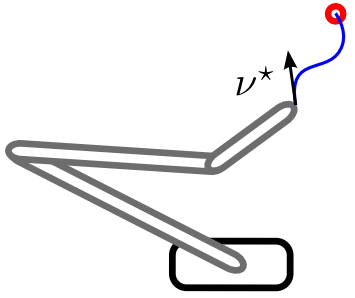


ξ: task space position

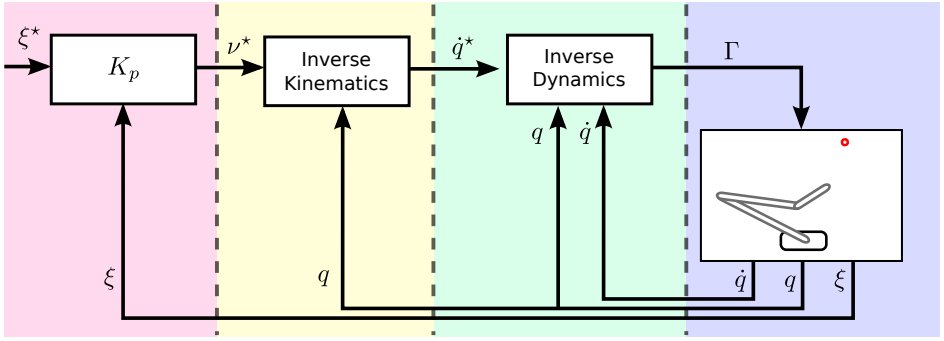
$\xi^\star$: desired task space position

$\nu^\star$: desired task space velocity





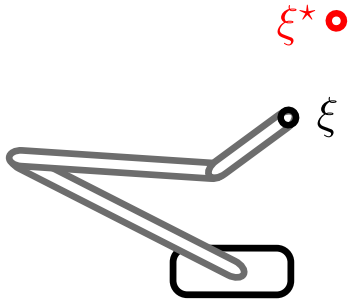
Planning: simplest implementation

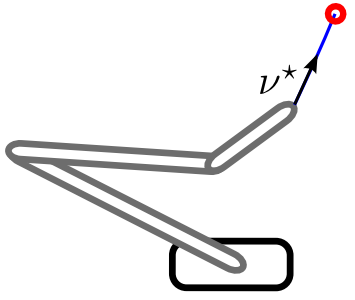


ξ: task space position

$\xi^\star$: desired task space position

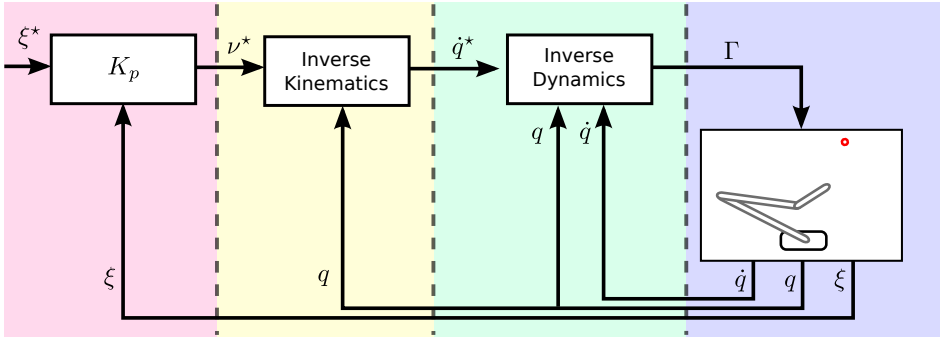
$\nu^\star$: desired task space velocity





Goal attractor: $\nu^\star = K_p\,(\xi^\star - \xi)$

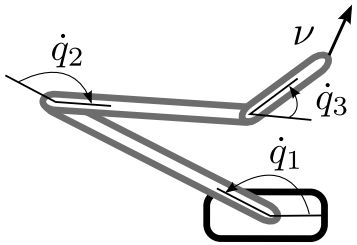
Inverse velocity kinematics



q: articular position

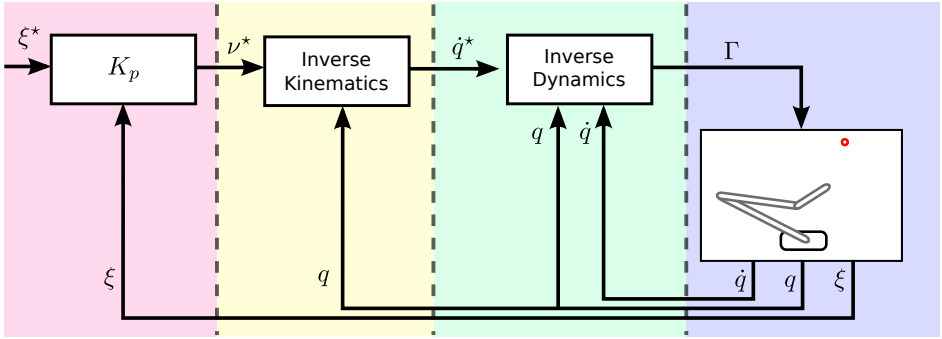
ν: task space velocity

$\dot{q}$: articular velocity



Velocity Kinematics: $\nu = J(q)\,\dot{q}$

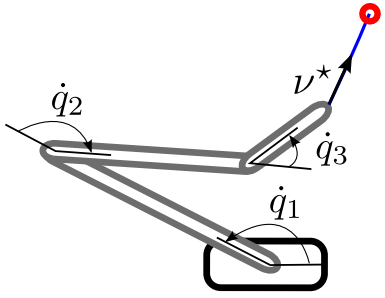
Inverse velocity kinematics



q: articular position

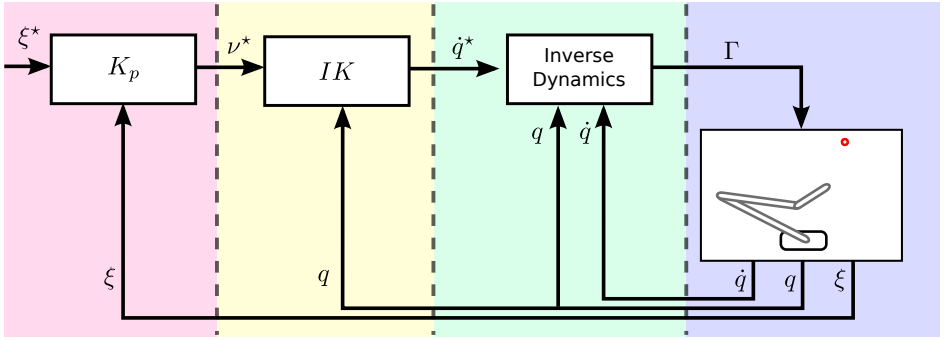
ν: task space velocity

$\dot{q}$: articular velocity



Inverse Velocity Kinematics: $\dot{q}^\star = J(q)^+ \nu^\star$

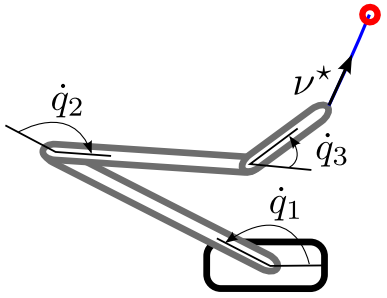
Inverse velocity kinematics



q: articular position

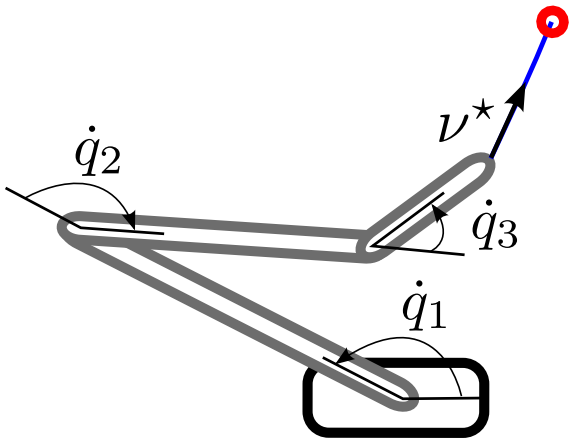
ν: task space velocity

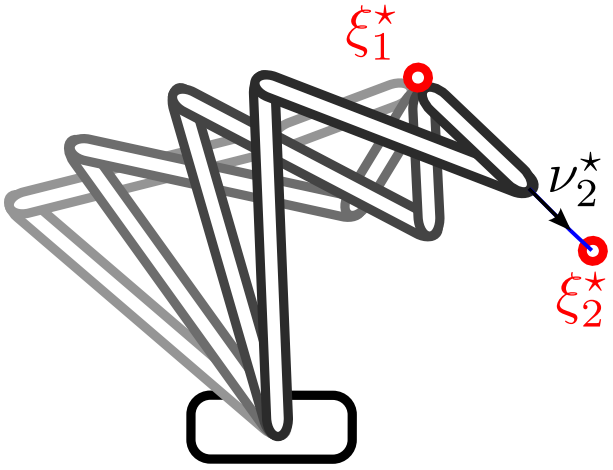
$\dot{q}$: articular velocity



Inverse Velocity Kinematics: $\dot{q}^\star = IVK\,(q, \nu^\star)$

Control redundancy





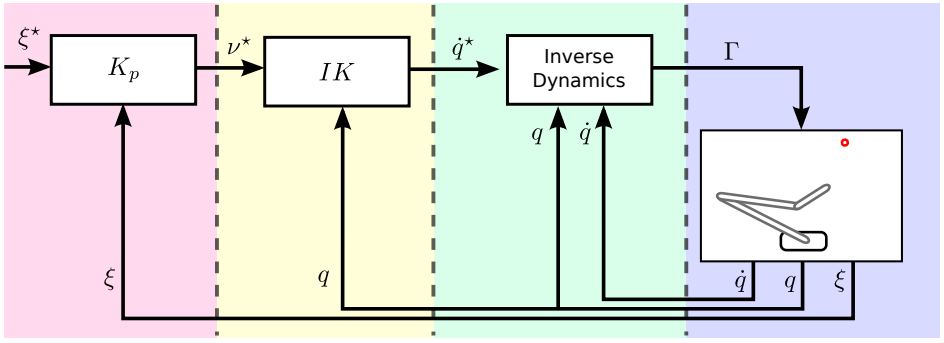
$$\dot{q}^\star = J(q)^+ \nu^\star$$

$$\dot{q}^\star = J_1(q)^+ \nu_1^\star + (J_2(q) P_{J_1})^+ \nu_2^\star$$

with $J^+ = J^t(JJ^t)^{-1}$ and $P_J = (I - J^+ J)$

- redundancy (short def.): "more actuated degrees of freedom than those necessary to realise a task"
- P_J is a projector onto the kernel of the Jacobian matrix J

Inverse Dynamics



Γ: torques

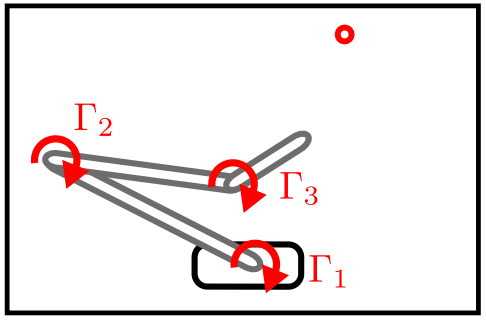
M: inertia matrix

b: Coriolis and centrifugal effects

g: gravity

ϵ: unmodeled effects

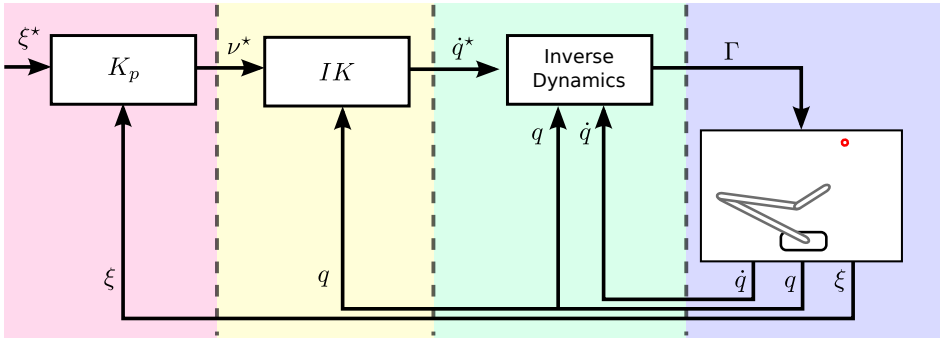
Γ^{ext}: external forces



Forward dynamics:

$$\ddot{q} = M^{-1}\left(\Gamma - b\left(q, \dot{q}\right) - g\left(q\right) - \epsilon\left(q, \dot{q}\right) + \Gamma^{ext}\right)$$

Inverse Dynamics



Γ: torques

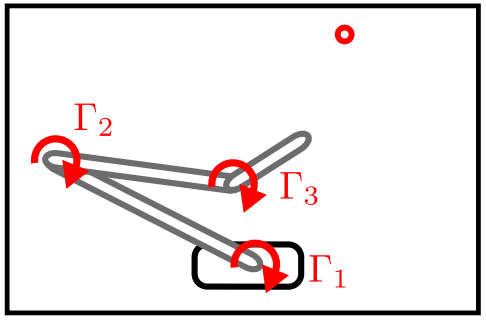
M: inertia matrix

b: Coriolis and centrifugal effects

g: gravity

ϵ: unmodeled effects

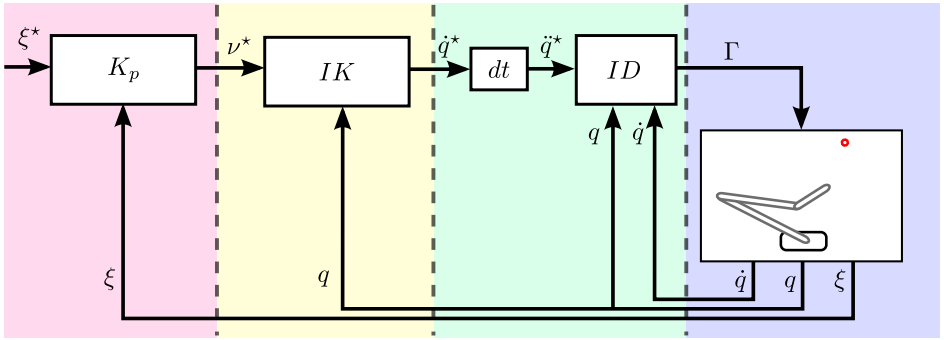
Γ^{ext}: external forces



Inverse dynamics:

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

Inverse Dynamics



Γ: torques

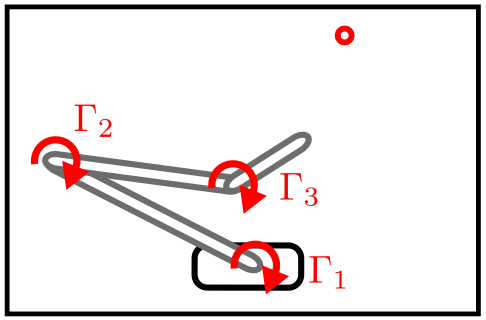
M: inertia matrix

b: Coriolis and centrifugal effects

g: gravity

ϵ: unmodeled effects

Γ^{ext}: external forces

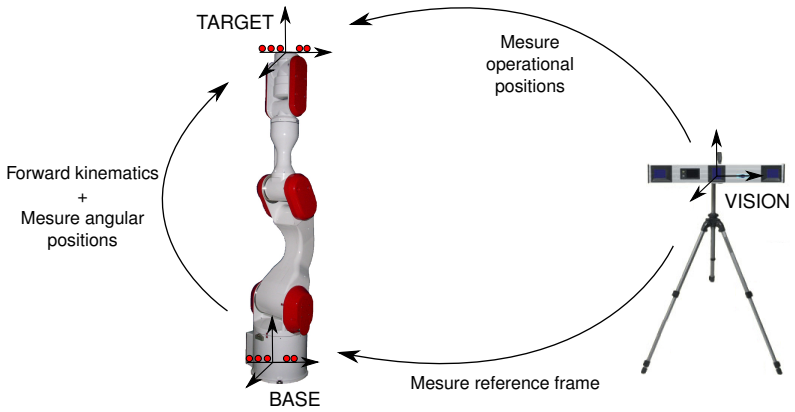


Inverse dynamics:

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

Identification

→ The control scheme presented here is an example of state-of-the-art result but requires a knowledge of the involved models.



Classical parametric identification:

+ least-squares minimization problem;
+ tuning of a parametered model;
+ suitable and robust for rigid-body systems;
- impossible to learn a non-modelized part;
- not suitable if the model structure evolves.

Why learning models for Robotics (from a roboticist point of view) ?

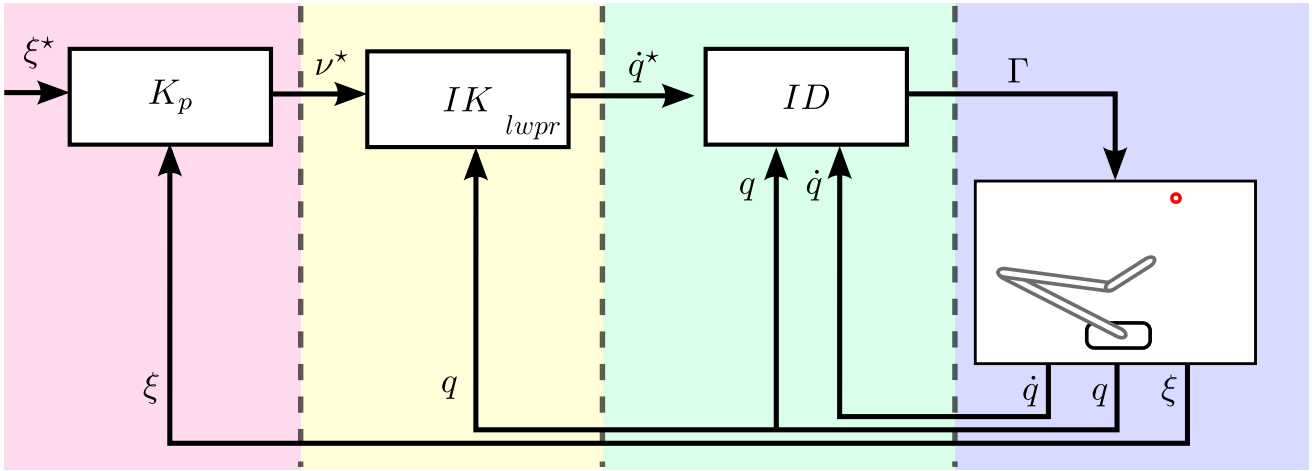
- Parametric identification approaches have limitations;
- Some phenomenon are hard to model and identify: friction, local mechanical compliance, etc.;
- Robots may, in the coming years, switch from rigid-body systems to structurally compliant systems;
- The models of the robot itself may be known but this is not true for its environment.

Why learning models for Robotics (from a roboticist point of view) ?

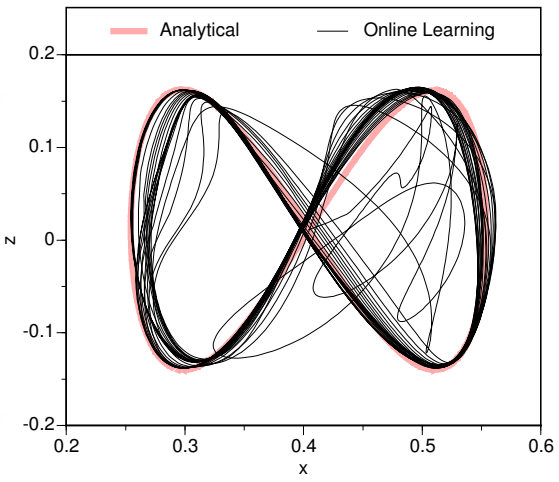
- Parametric identification approaches have limitations;
- Some phenomenon are hard to model and identify: friction, local mechanical compliance, etc.;
- Robots may, in the coming years, switch from rigid-body systems to structurally compliant systems;
- The models of the robot itself may be known but this is not true for its environment.

Does it work ?

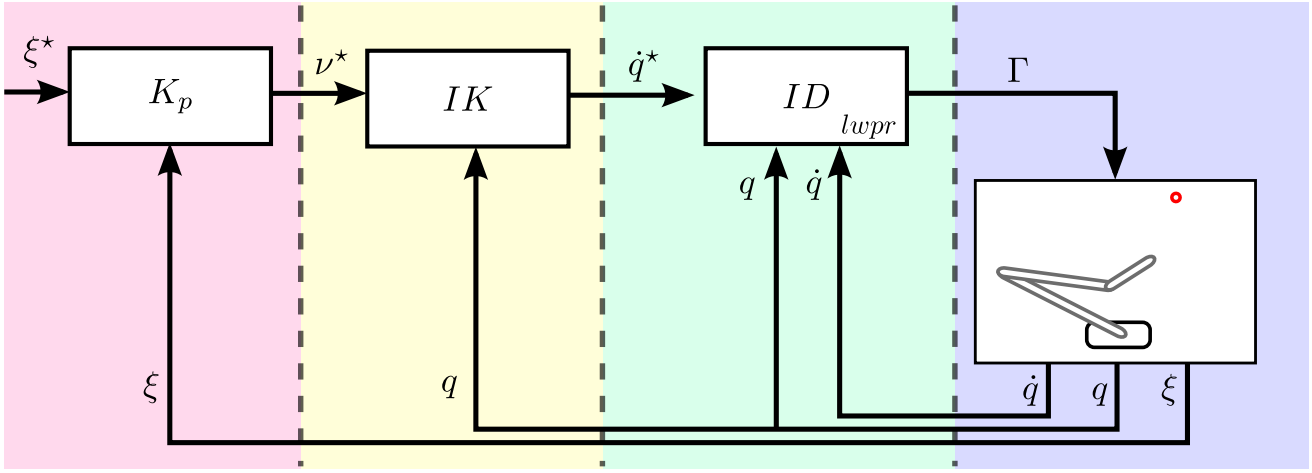
Learning inverse velocity kinematics



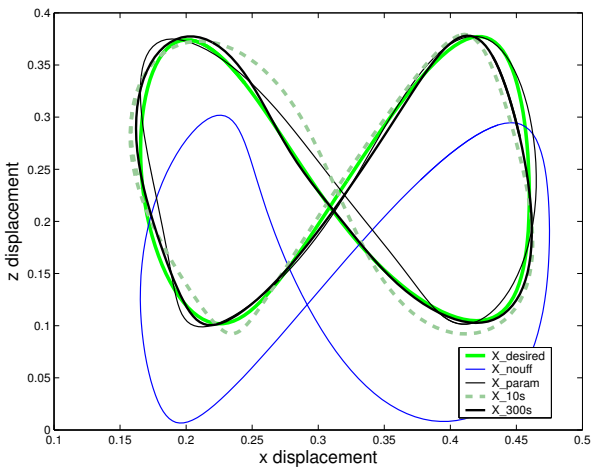
- D'Souza et al. (1) learn the inverse velocity kinematics model
- The model is learned along an task space trajectory
- Input dimension: $dim(xi + q) = 29$
- Output dimension: $dim(\dot{q}) = 26$



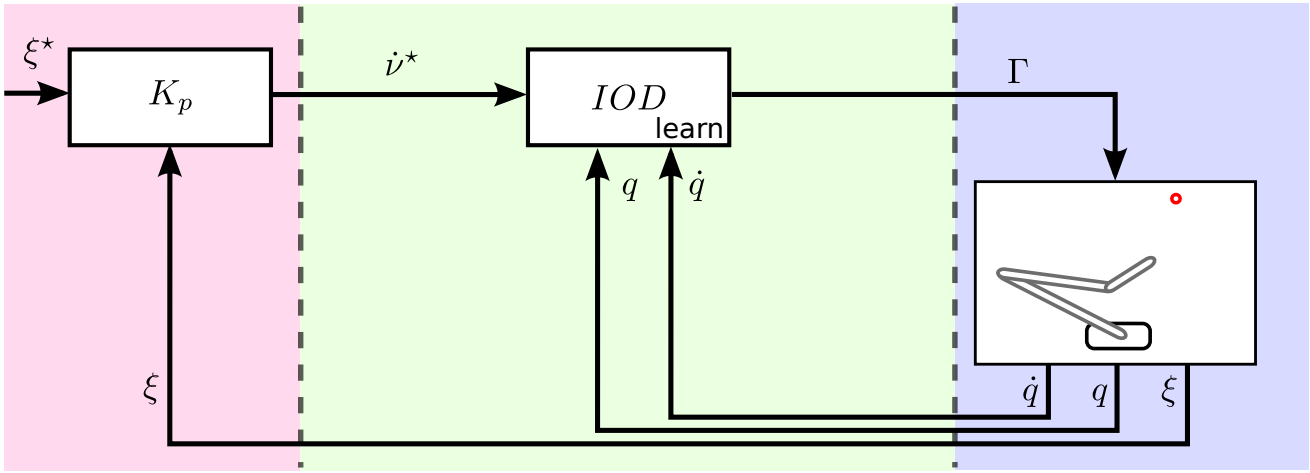
Learning inverse dynamics



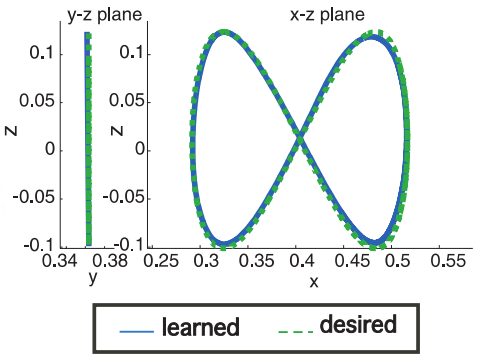
- Schaal et al. (2) learn the inverse dynamics model
- The model is learned along an task space trajectory
- Input dimension: $dim(q + \dot{q} + \ddot{q}) = 90$
- Output dimension: $dim(\Gamma) = 30$



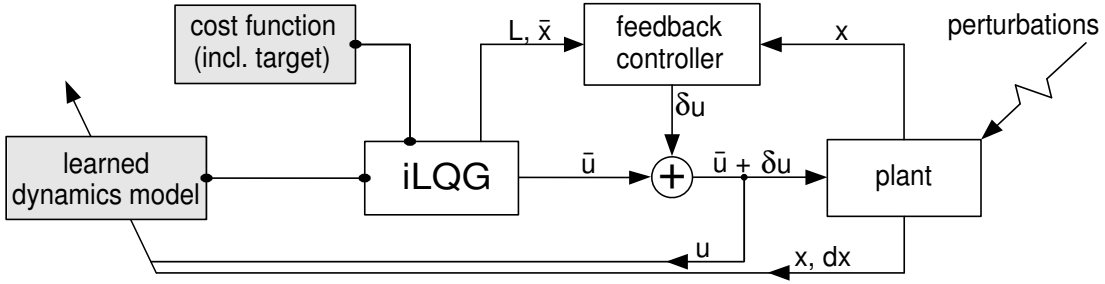
Learning inverse task space dynamics



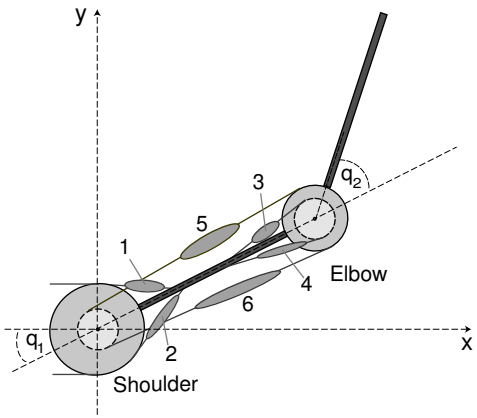
- Peters and Schaal (3) learn inverse dynamics in the task space space.
- The model is learned along an task space trajectory.
- Input dimension: 7+7+3=17. Output dimension: 7.



Optimal control with dynamics learned with LWPR



- Mitrovic et al. (4) learn inverse dynamics.
- The 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



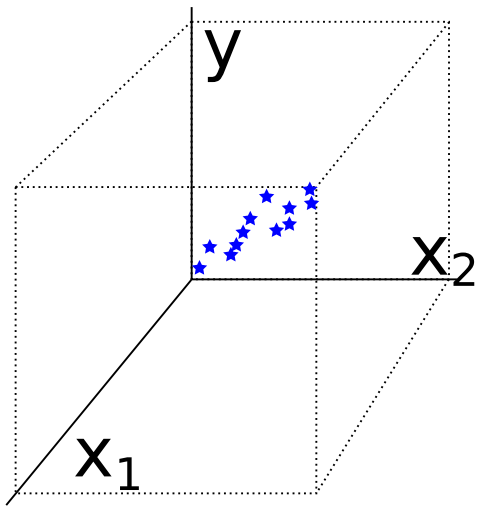
How does it work ?

Linear functions approximation:

Least Squares, Recursive Least Squares and Partial Least Squares

Linear functions approximation:

Least Squares, Recursive Least Squares and Partial Least Squares

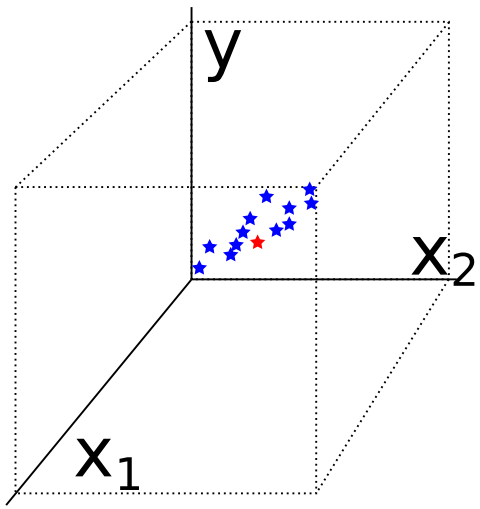


Least Squares

- Least Squares: Not incremental, need to save data and run a batch identification

Linear functions approximation:

Least Squares, Recursive Least Squares and Partial Least Squares

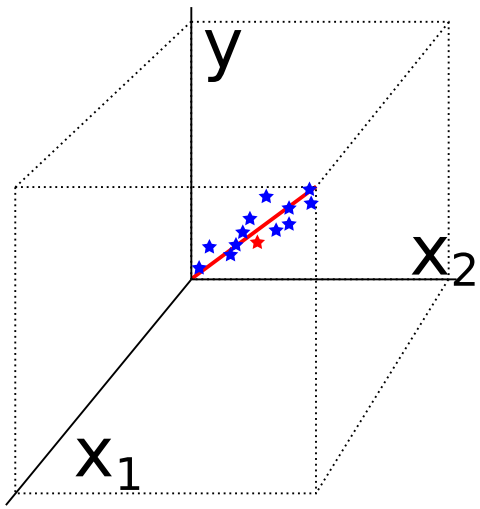


Least Squares

- Least Squares: Not incremental, need to save data and run a batch identification

Linear functions approximation:

Least Squares, Recursive Least Squares and Partial Least Squares

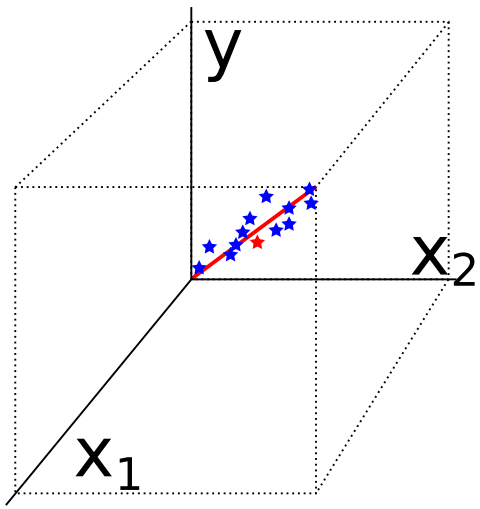


Least Squares

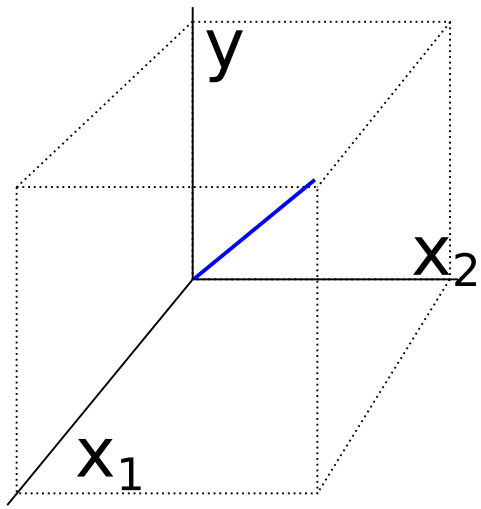
- Least Squares: Not incremental, need to save data and run a batch identification

Linear functions approximation:

Least Squares, Recursive Least Squares and Partial Least Squares



Least Squares



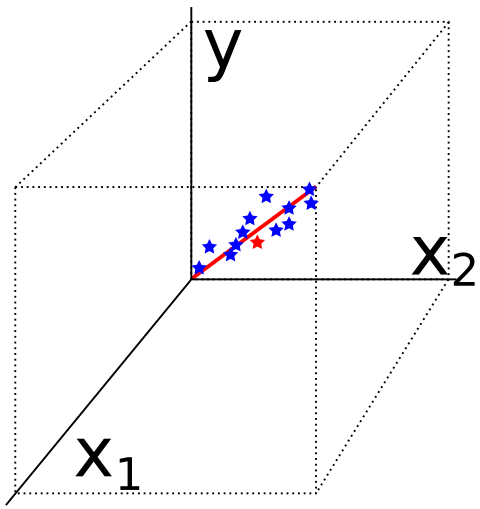
Recursive Least Squares

- Least Squares: Not incremental, need to save data and run a batch identification
- Recursive Least Squares: Incremental but diverges with linearly dependent inputs

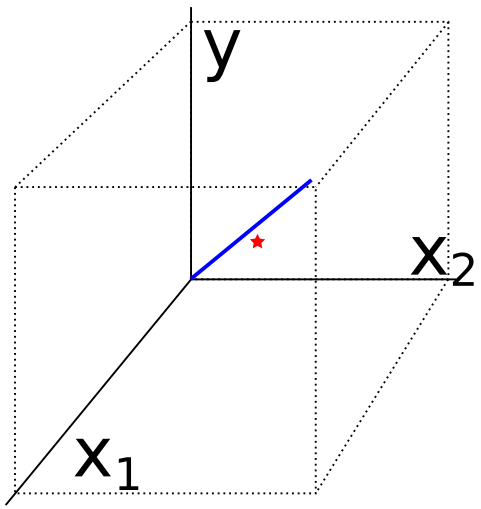
How does it work ?

Linear functions approximation:

Least Squares, Recursive Least Squares and Partial Least Squares



Least Squares



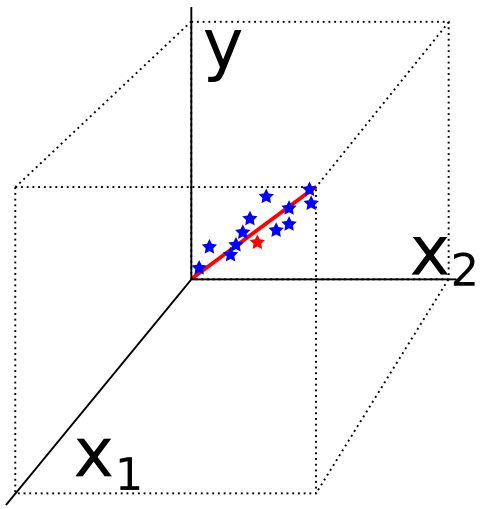
Recursive Least Squares

- Least Squares: Not incremental, need to save data and run a batch identification
- Recursive Least Squares: Incremental but diverges with linearly dependent inputs

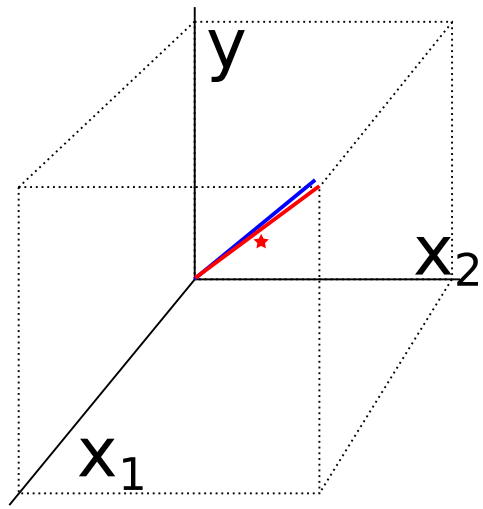
How does it work ?

Linear functions approximation:

Least Squares, Recursive Least Squares and Partial Least Squares



Least Squares

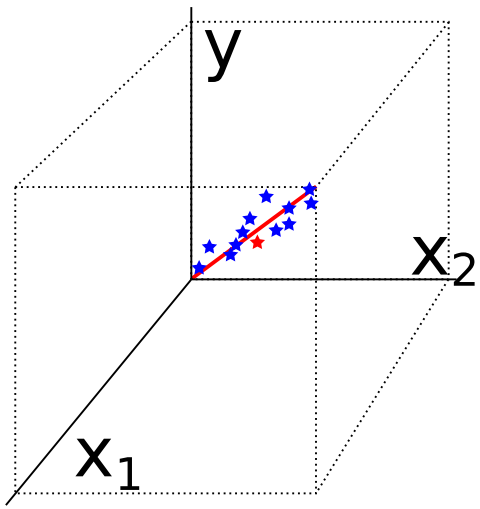


Recursive Least Squares

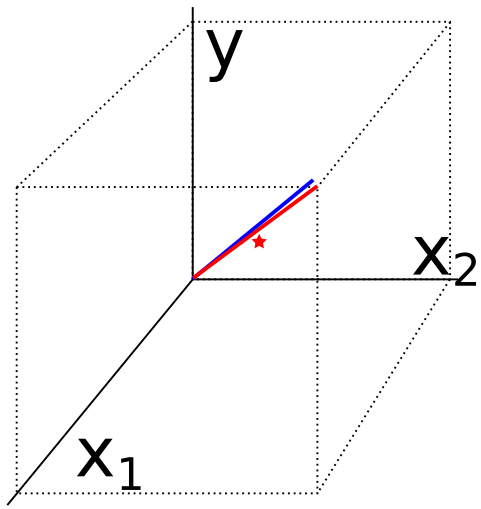
- Least Squares: Not incremental, need to save data and run a batch identification
- Recursive Least Squares: Incremental but diverges with linearly dependent inputs

Linear functions approximation:

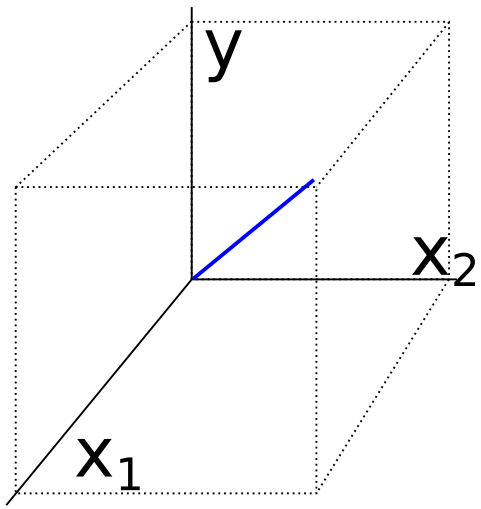
Least Squares, Recursive Least Squares and Partial Least Squares



Least Squares



Recursive Least Squares

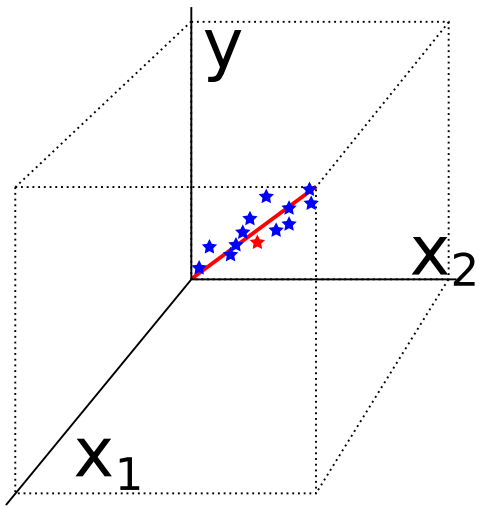


Partial Least Squares

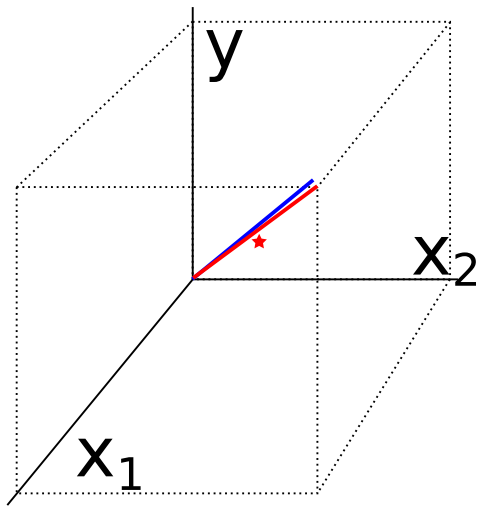
- Least Squares: Not incremental, need to save data and run a batch identification
- Recursive Least Squares: Incremental but diverges with linearly dependent inputs
- Partial Least Squares: Projections to focus on relevant inputs

Linear functions approximation:

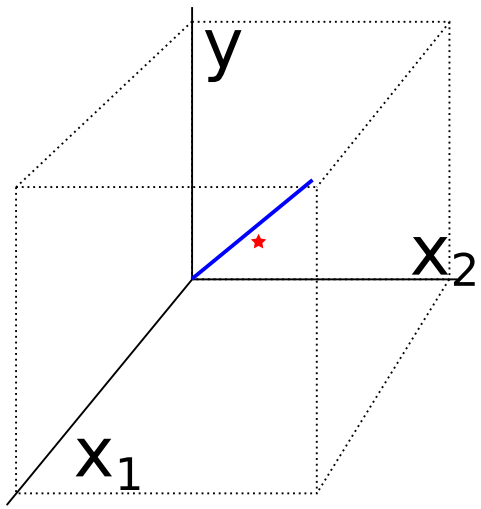
Least Squares, Recursive Least Squares and Partial Least Squares



Least Squares



Recursive Least Squares

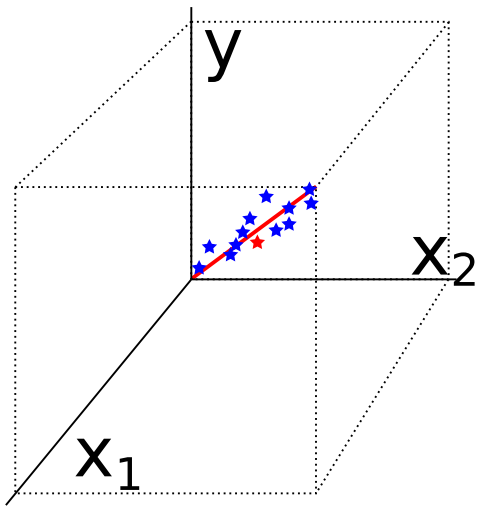


Partial Least Squares

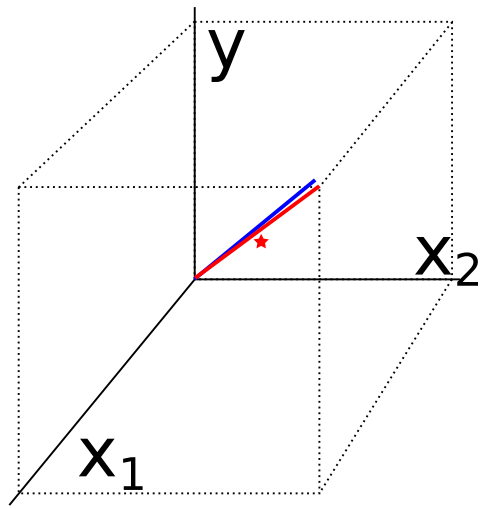
- Least Squares: Not incremental, need to save data and run a batch identification
- Recursive Least Squares: Incremental but diverges with linearly dependent inputs
- Partial Least Squares: Projections to focus on relevant inputs

Linear functions approximation:

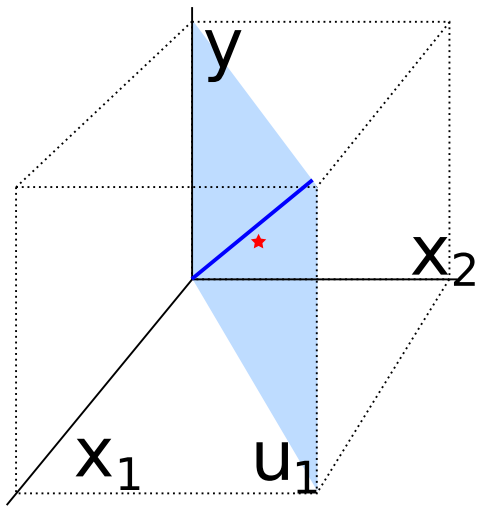
Least Squares, Recursive Least Squares and Partial Least Squares



Least Squares



Recursive Least Squares

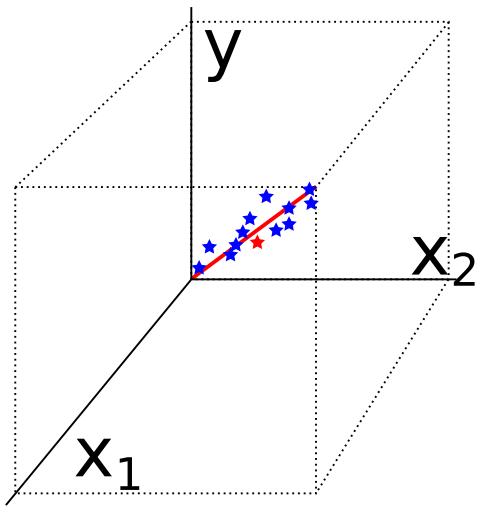


Partial Least Squares

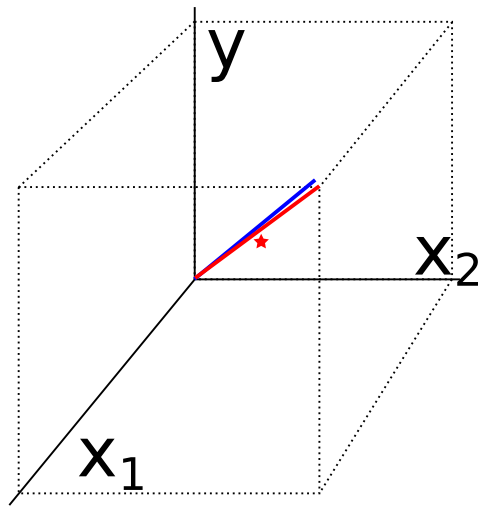
- Least Squares: Not incremental, need to save data and run a batch identification
- Recursive Least Squares: Incremental but diverges with linearly dependent inputs
- Partial Least Squares: Projections to focus on relevant inputs

Linear functions approximation:

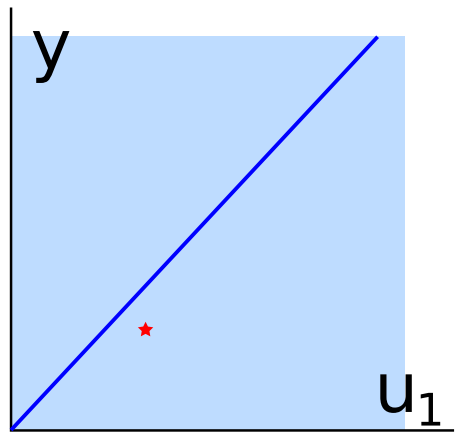
Least Squares, Recursive Least Squares and Partial Least Squares



Least Squares



Recursive Least Squares



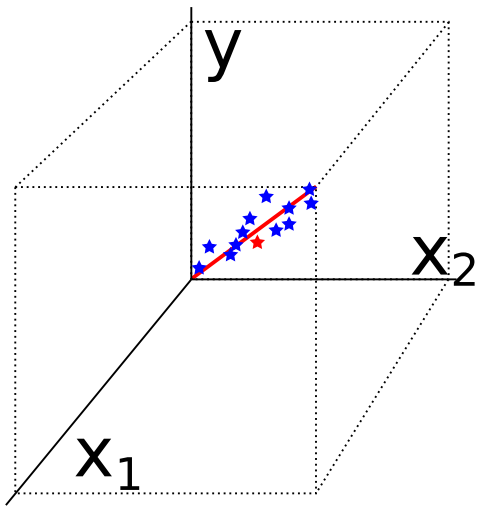
Partial Least Squares

- Least Squares: Not incremental, need to save data and run a batch identification
- Recursive Least Squares: Incremental but diverges with linearly dependent inputs
- Partial Least Squares: Projections to focus on relevant inputs

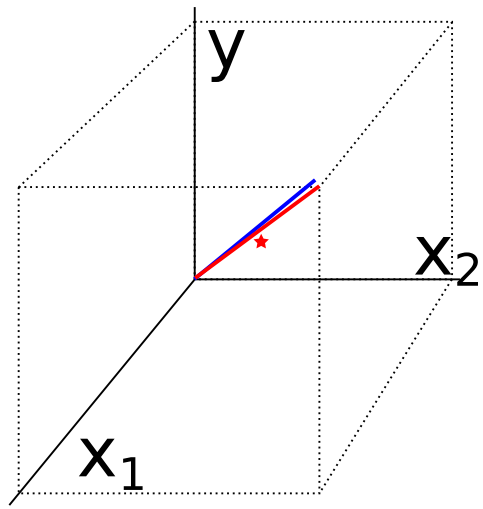
How does it work ?

Linear functions approximation:

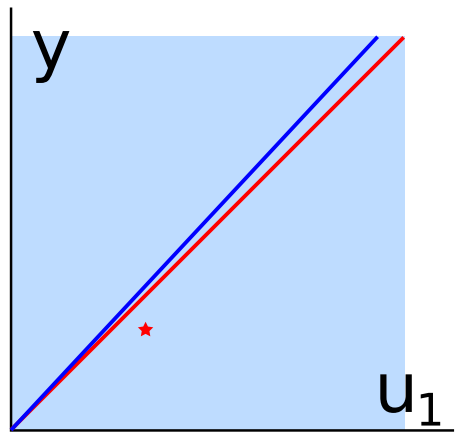
Least Squares, Recursive Least Squares and Partial Least Squares



Least Squares



Recursive Least Squares

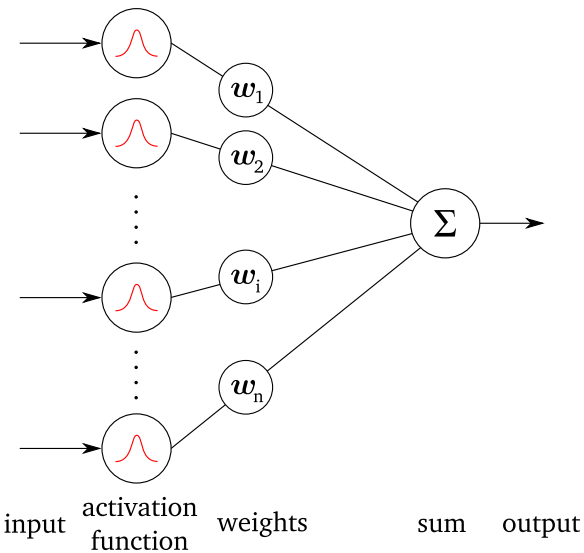


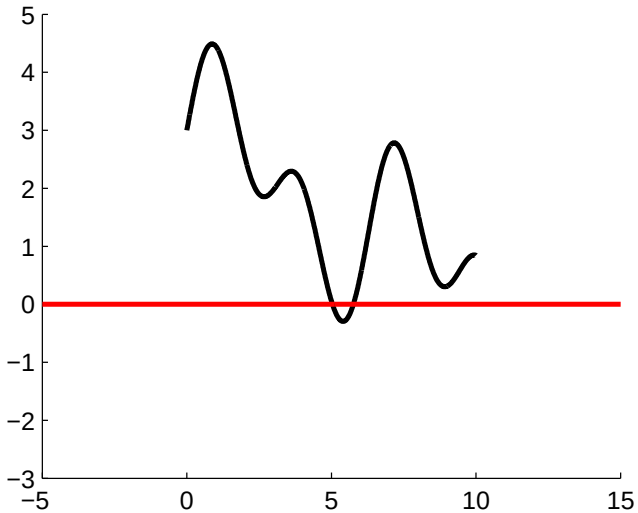
Partial Least Squares

- Least Squares: Not incremental, need to save data and run a batch identification
- Recursive Least Squares: Incremental but diverges with linearly dependent inputs
- Partial Least Squares: Projections to focus on relevant inputs

Nonlinear functions approximations using nonlinear elementary functions:

Radial Basis Function Networks

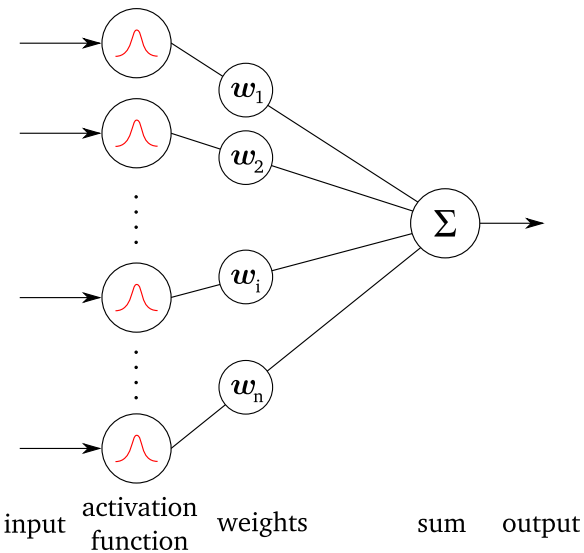


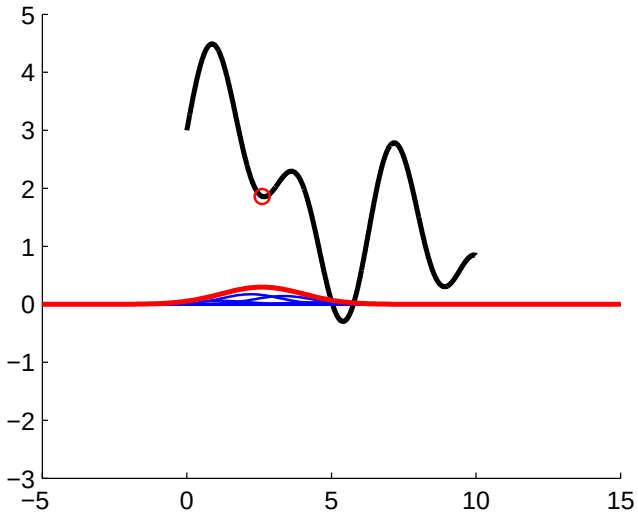


$$\hat{y}(x) = \sum_{i=1}^{N} w_i e^{-\beta_i \|x - c_i\|^2}$$

Nonlinear functions approximations using nonlinear elementary functions:

Radial Basis Function Networks

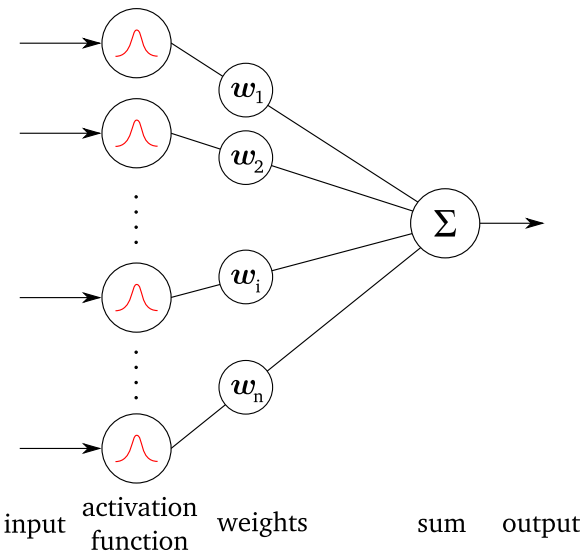


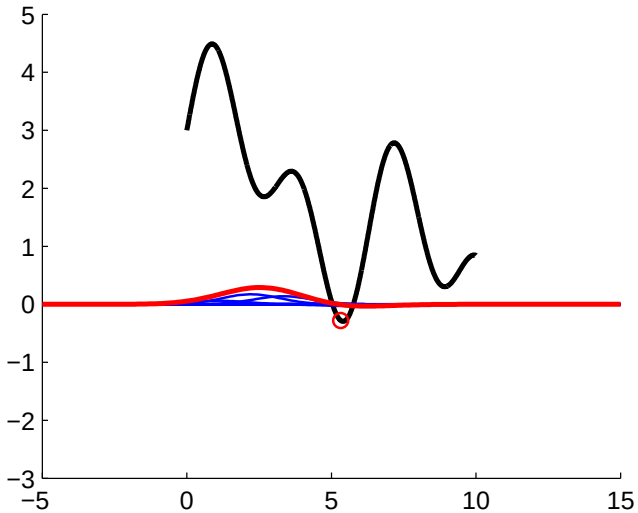


$$\hat{y}(x) = \sum_{i=1}^{N} w_i e^{-\beta_i \|x - c_i\|^2}$$

Nonlinear functions approximations using nonlinear elementary functions:

Radial Basis Function Networks

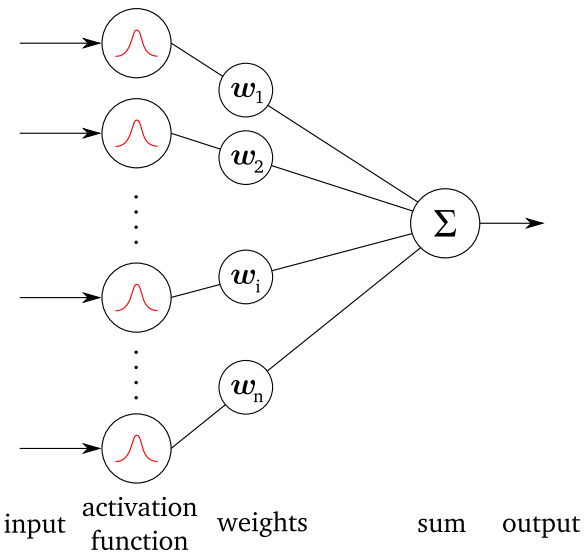


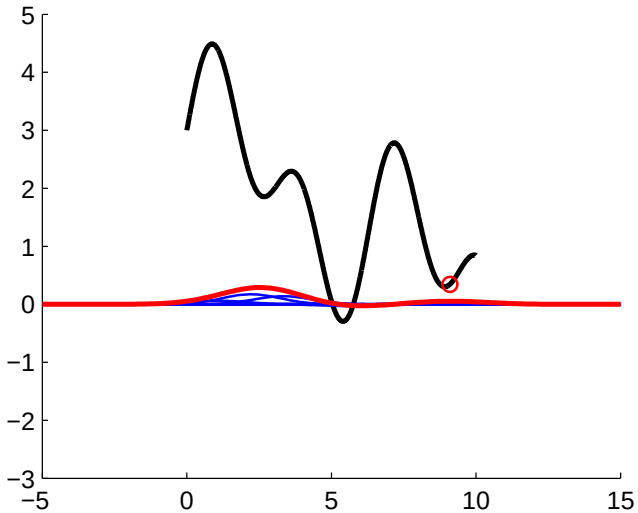


$$\hat{y}(x) = \sum_{i=1}^{N} w_i e^{-\beta_i \|x - c_i\|^2}$$

Nonlinear functions approximations using nonlinear elementary functions:

Radial Basis Function Networks

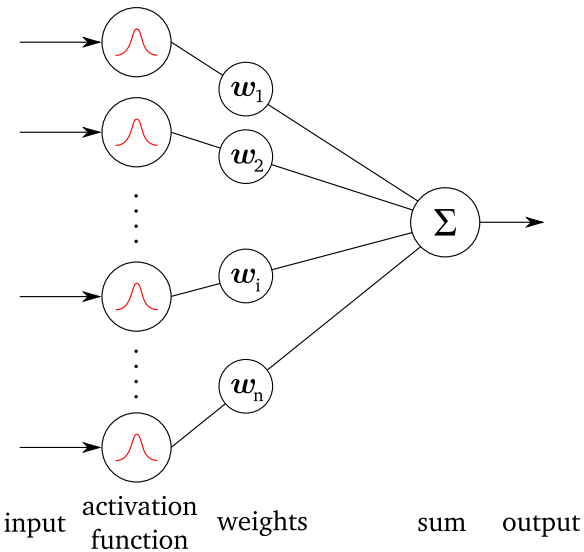


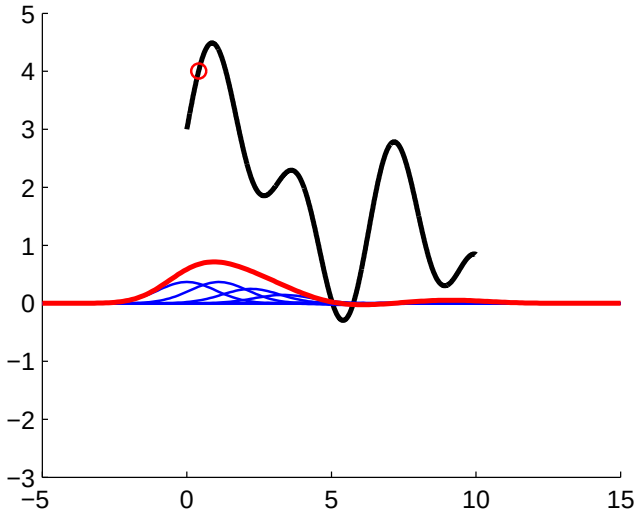


$$\hat{y}(x) = \sum_{i=1}^{N} w_i e^{-\beta_i \|x - c_i\|^2}$$

Nonlinear functions approximations using nonlinear elementary functions:

Radial Basis Function Networks

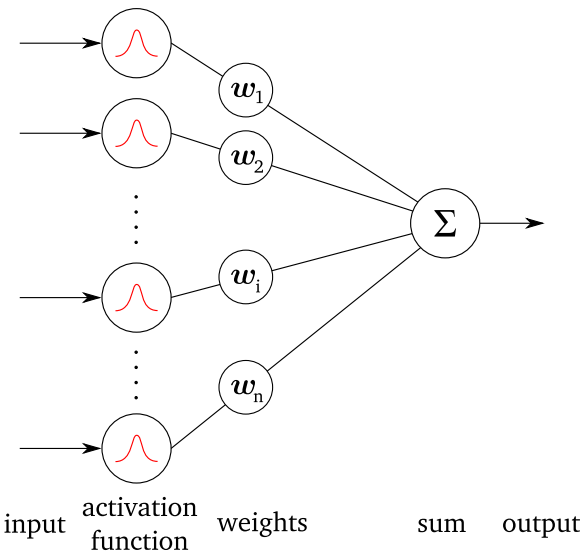


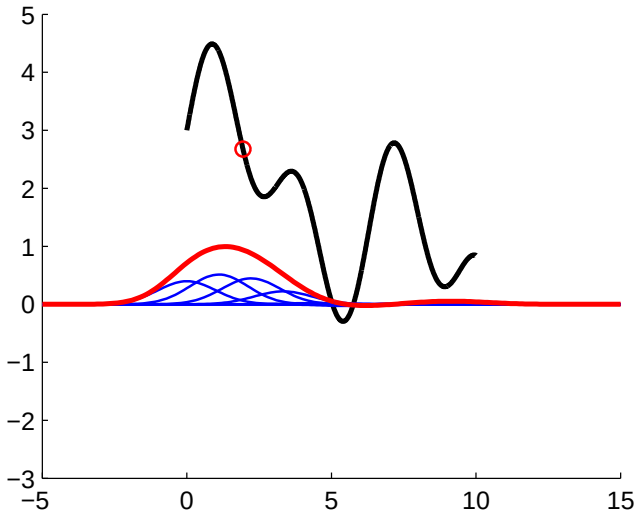


$$\hat{y}(x) = \sum_{i=1}^{N} w_i e^{-\beta_i \|x - c_i\|^2}$$

Nonlinear functions approximations using nonlinear elementary functions:

Radial Basis Function Networks

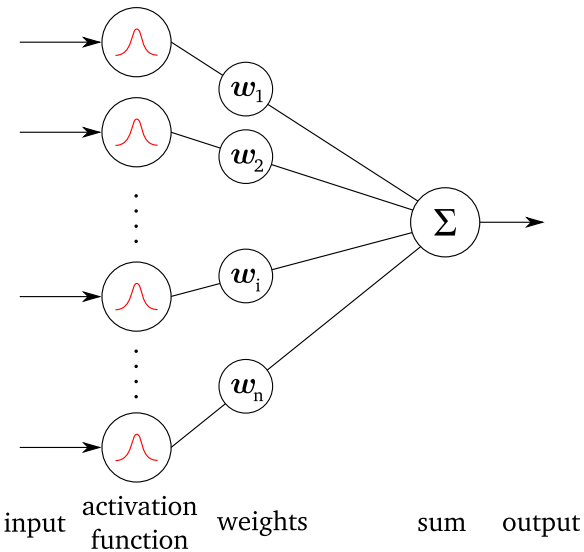


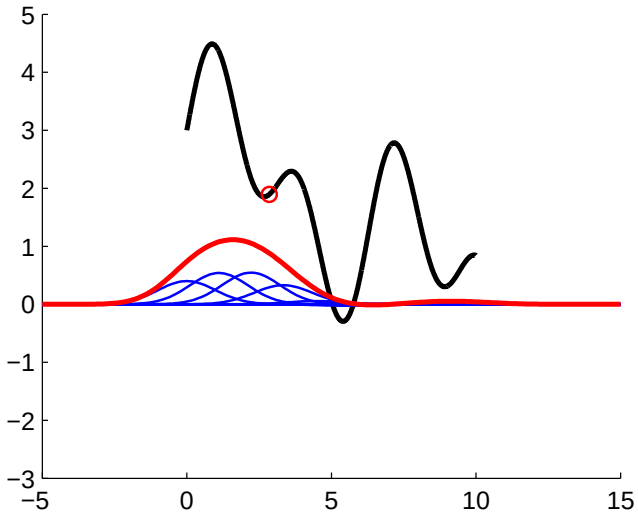


$$\hat{y}(x) = \sum_{i=1}^{N} w_i e^{-\beta_i \|x - c_i\|^2}$$

Nonlinear functions approximations using nonlinear elementary functions:

Radial Basis Function Networks

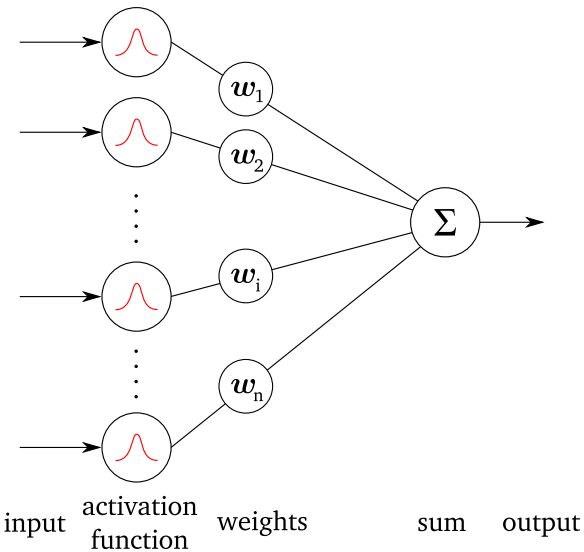


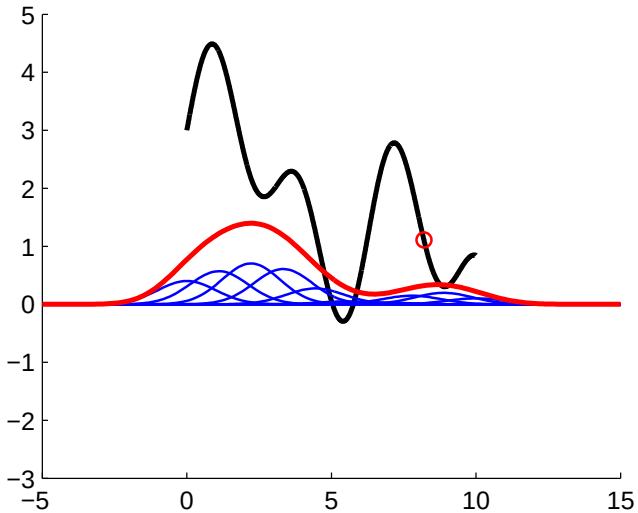


$$\hat{y}(x) = \sum_{i=1}^{N} w_i e^{-\beta_i \|x - c_i\|^2}$$

Nonlinear functions approximations using nonlinear elementary functions:

Radial Basis Function Networks

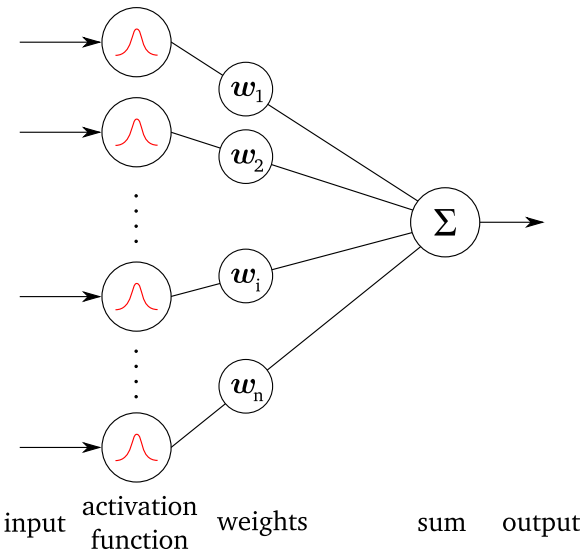


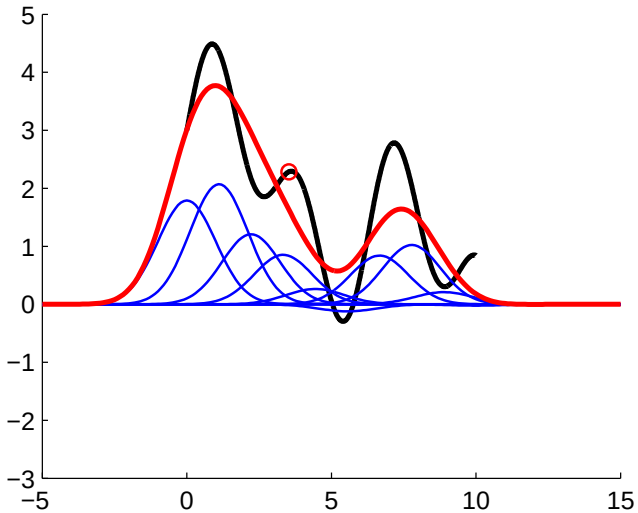


$$\hat{y}(x) = \sum_{i=1}^{N} w_i e^{-\beta_i \|x - c_i\|^2}$$

Nonlinear functions approximations using nonlinear elementary functions:

Radial Basis Function Networks

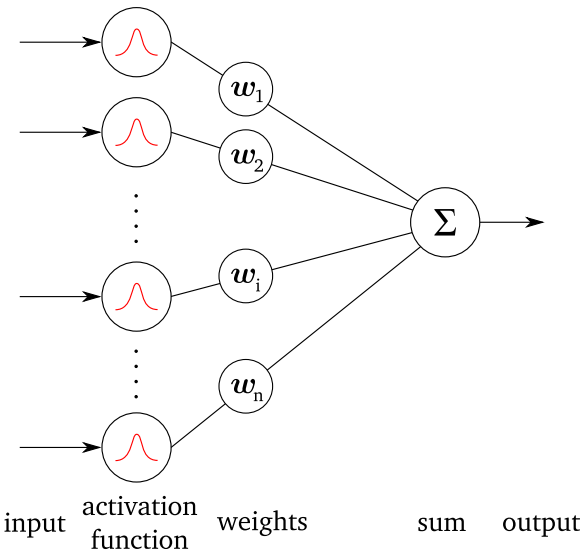


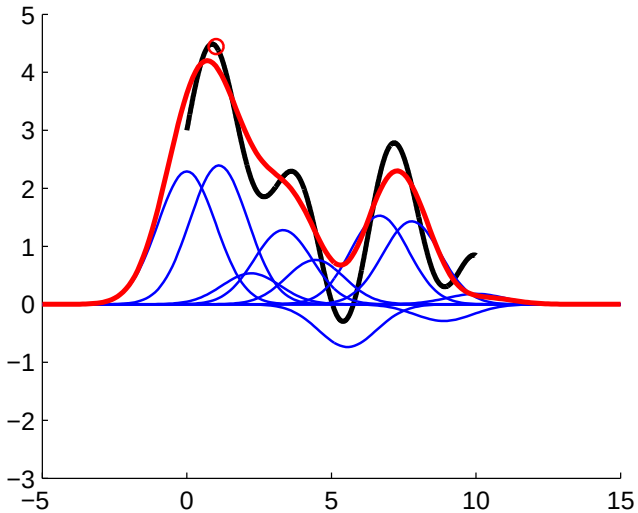


$$\hat{y}(x) = \sum_{i=1}^{N} w_i e^{-\beta_i \|x - c_i\|^2}$$

Nonlinear functions approximations using nonlinear elementary functions:

Radial Basis Function Networks

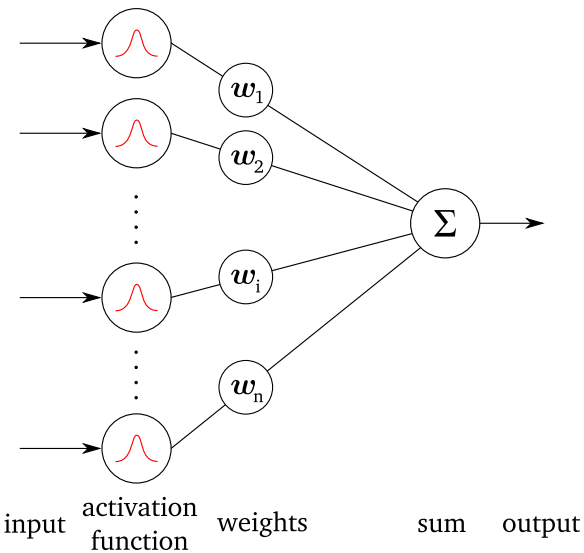


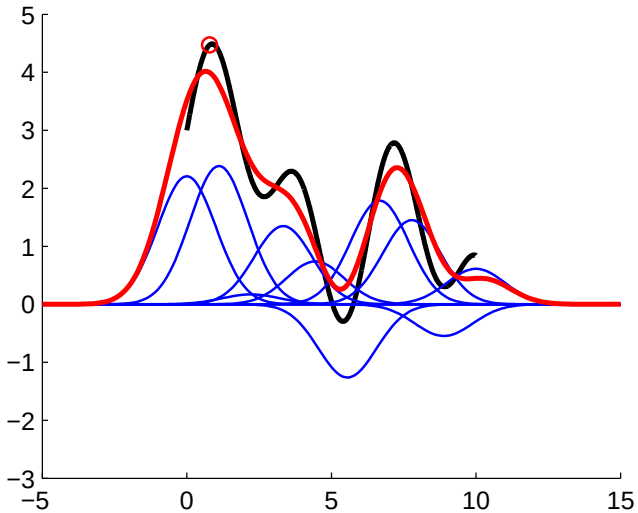


$$\hat{y}(x) = \sum_{i=1}^{N} w_i e^{-\beta_i \|x - c_i\|^2}$$

Nonlinear functions approximations using nonlinear elementary functions:

Radial Basis Function Networks



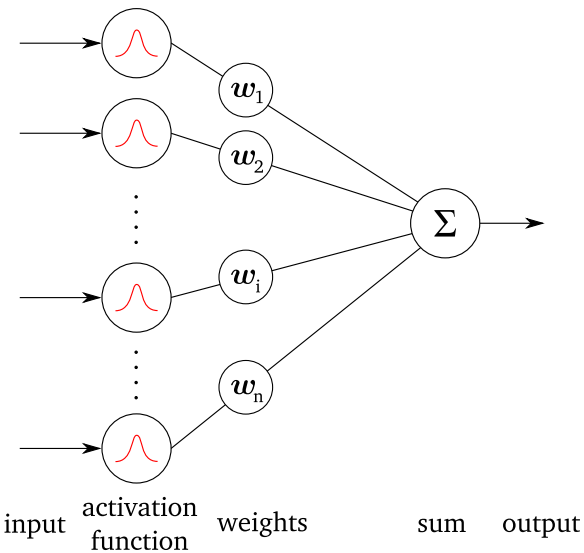


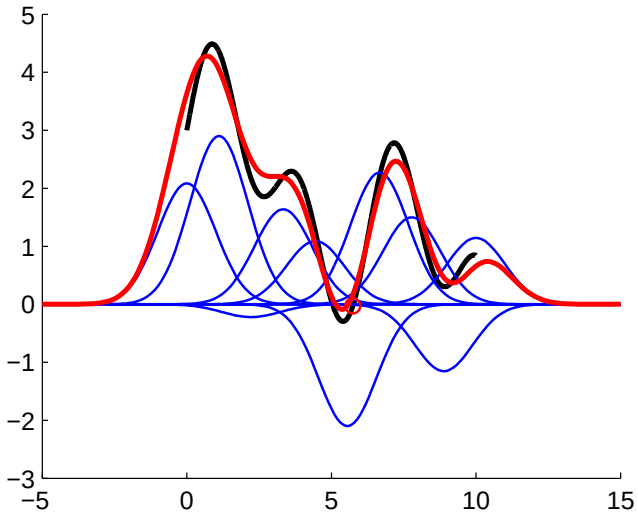
$$\hat{y}(x) = \sum_{i=1}^{N} w_i e^{-\beta_i \|x - c_i\|^2}$$

How does it work ?

Nonlinear functions approximations using nonlinear elementary functions:

Radial Basis Function Networks

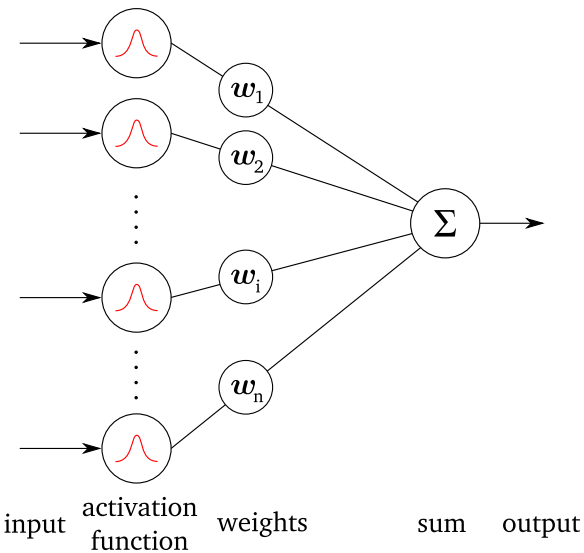


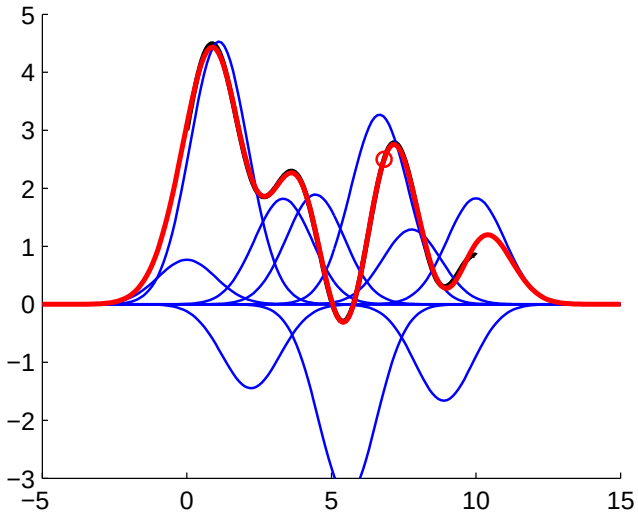


$$\hat{y}(x) = \sum_{i=1}^{N} w_i e^{-\beta_i \|x - c_i\|^2}$$

Nonlinear functions approximations using nonlinear elementary functions:

Radial Basis Function Networks

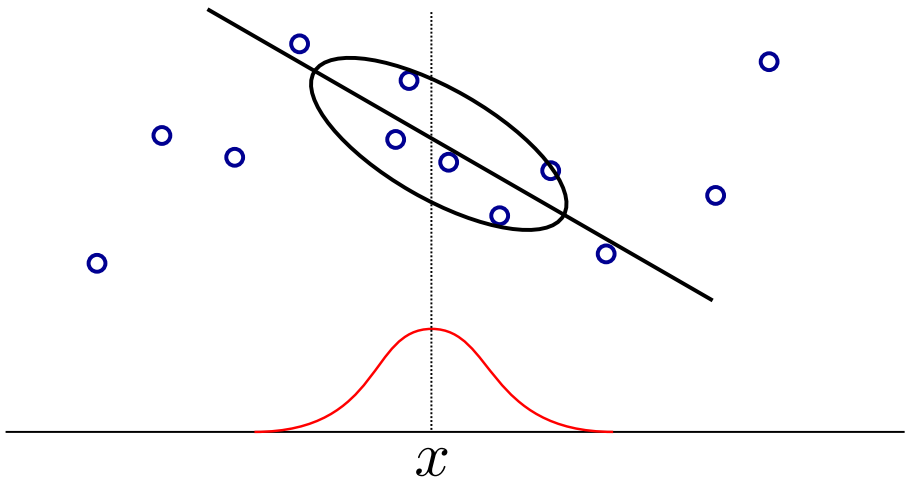


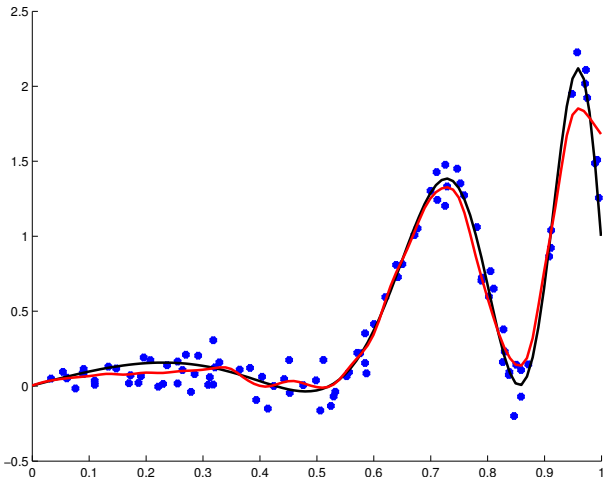


$$\hat{y}(x) = \sum_{i=1}^{N} w_i e^{-\beta_i \|x - c_i\|^2}$$

Nonlinear functions approximation using linear elementary functions:

Locally Weighted Regression

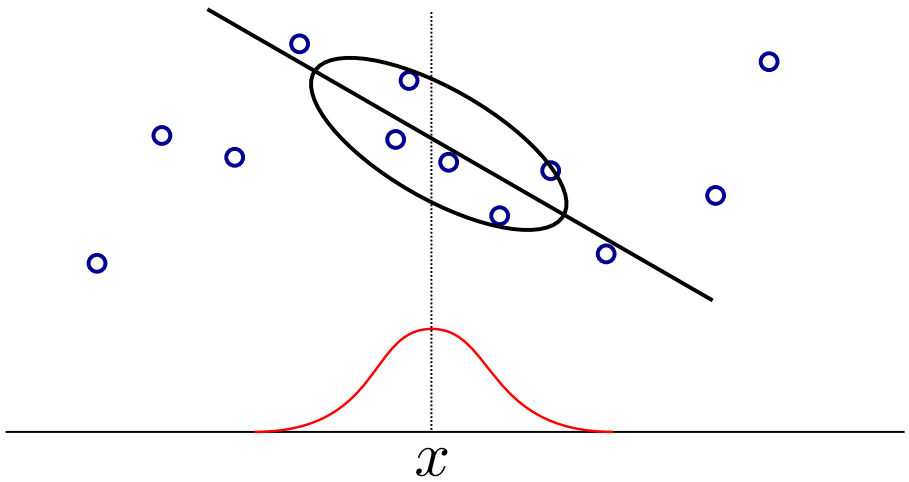


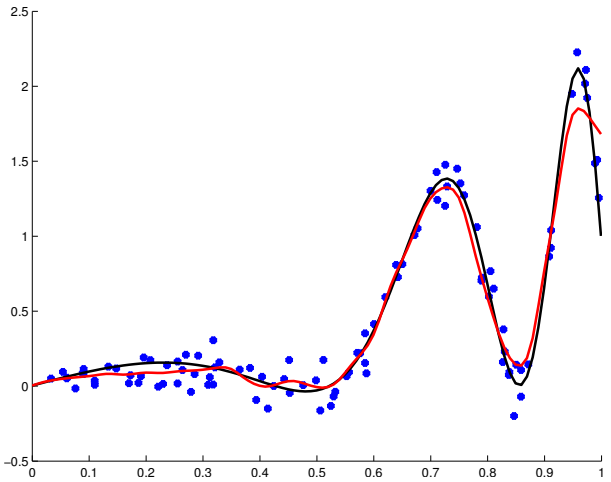


- $\phi_k = e^{-\frac{1}{2}(x_k - x)\sigma(x_k - x)}$ where $k \in [1\ q]$ span the number of saved points

Nonlinear functions approximation using linear elementary functions:

Locally Weighted Regression

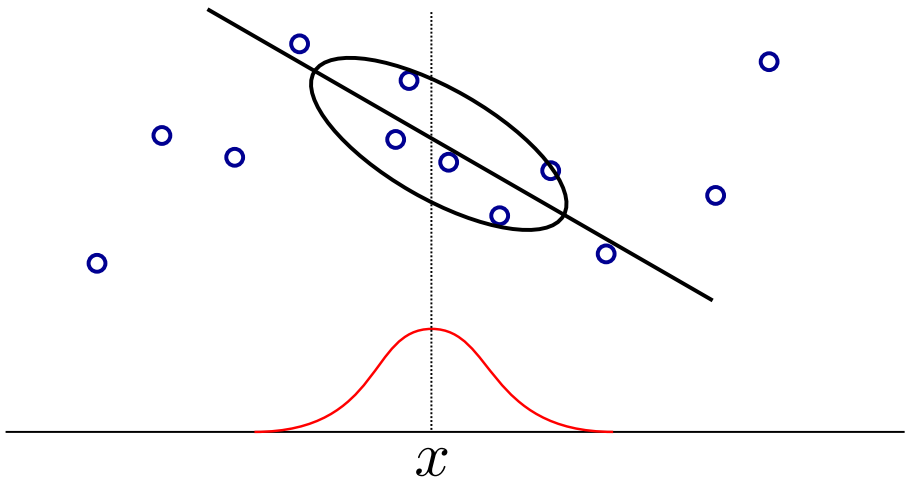


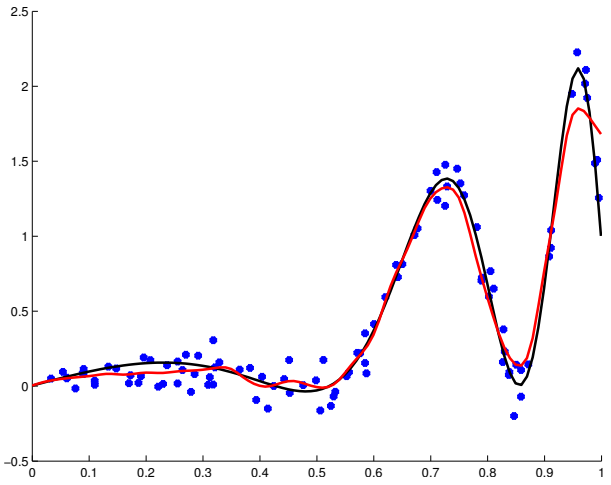


- $\phi_k = e^{-\frac{1}{2}(x_k - x)\sigma(x_k - x)}$ where $k \in [1\ q]$ span the number of saved points
- $\beta = X^{\Phi +} Y$ where $X = [x_1 \cdots x_q]$, $Y = [y_1 \cdots y_q]$ and $\Phi = [\phi_1 \cdots \phi_q]$.

Nonlinear functions approximation using linear elementary functions:

Locally Weighted Regression





- $\phi_k = e^{-\frac{1}{2}(x_k - x)\sigma(x_k - x)}$ where $k \in [1\ q]$ span the number of saved points
- $\beta = X^{\Phi+} Y$ where $X = [x_1 \cdots x_q]$, $Y = [y_1 \cdots y_q]$ and $\Phi = [\phi_1 \cdots \phi_q]$.
- $\hat{y} = \beta x$

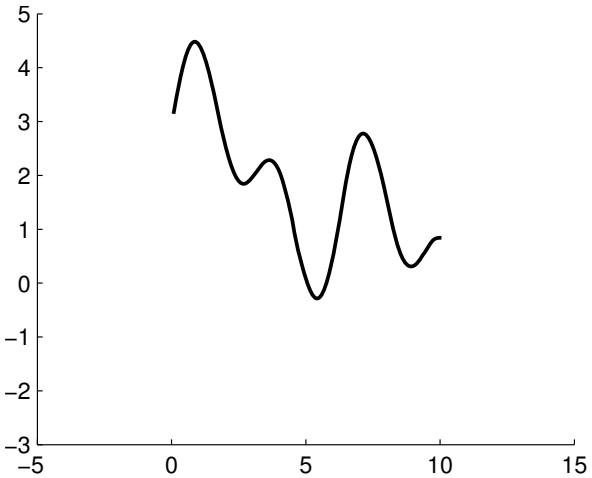
Locally Weighted Projection Regression: Definition [(6), (1), (7)]

Locally Weighted Projection Regression: Definition [(6), (1), (7)]

- Incremental algorithm which approximates non-linear functions with huge input space taking into account only relevant information
- LWPR is based on Locally Weighted Regression and Partial Least Squares

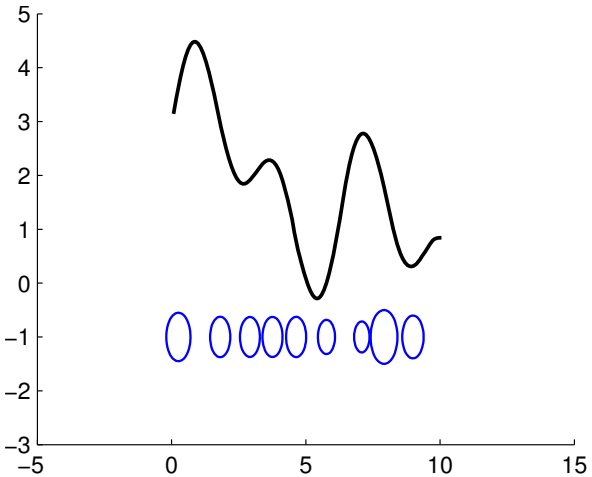
Locally Weighted Projection Regression: Definition [(6), (1), (7)]

- Incremental algorithm which approximates non-linear functions with huge input space taking into account only relevant information
- LWPR is based on Locally Weighted Regression and Partial Least Squares



Locally Weighted Projection Regression: Definition [(6), (1), (7)]

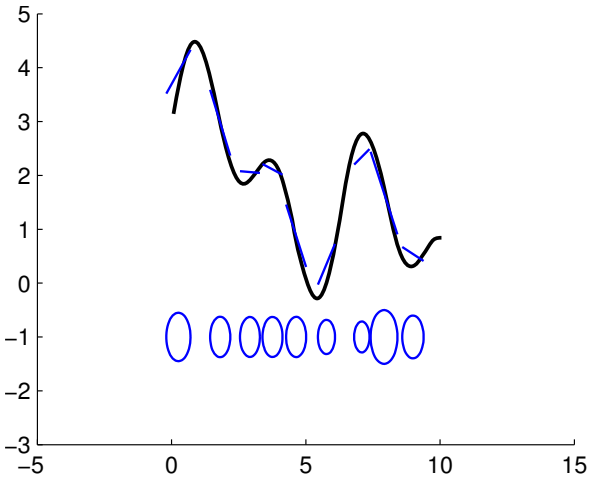
- Incremental algorithm which approximates non-linear functions with huge input space taking into account only relevant information
- LWPR is based on Locally Weighted Regression and Partial Least Squares



- $\phi_i = e^{-\frac{1}{2}(x-c_i)D_i(x-c_i)}$

Locally Weighted Projection Regression: Definition [(6), (1), (7)]

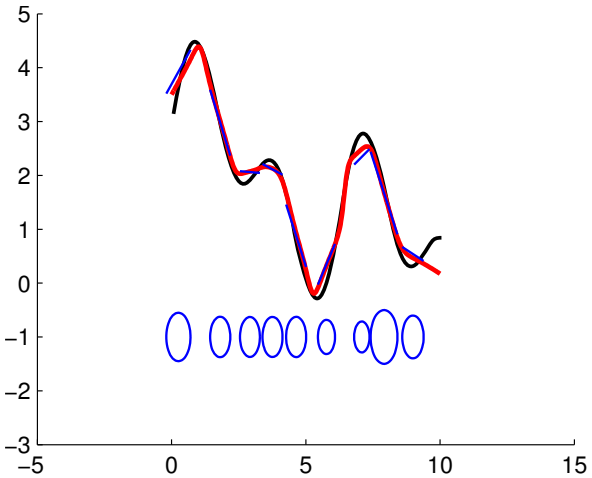
- Incremental algorithm which approximates non-linear functions with huge input space taking into account only relevant information
- LWPR is based on Locally Weighted Regression and Partial Least Squares



- $\phi_i = e^{-\frac{1}{2}(x-c_i)D_i(x-c_i)}$
- $y_i = \beta_i x$

Locally Weighted Projection Regression: Definition [(6), (1), (7)]

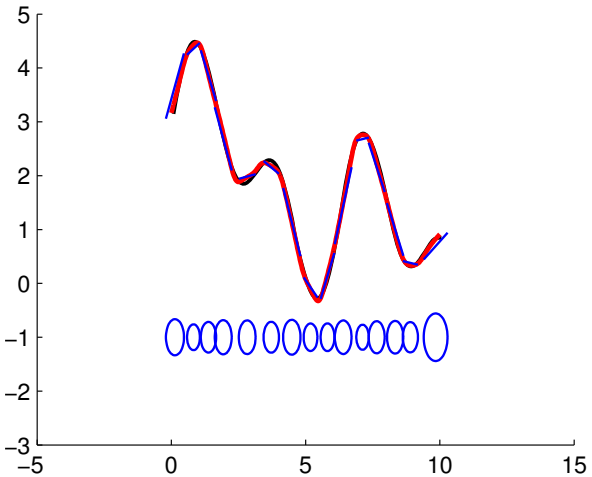
- Incremental algorithm which approximates non-linear functions with huge input space taking into account only relevant information
- LWPR is based on Locally Weighted Regression and Partial Least Squares



- $\phi_i = e^{-\frac{1}{2}(x-c_i)D_i(x-c_i)}$
- $y_i = \beta_i x$
- $\hat{y} = \frac{\sum_{i=1}^{N} \phi_i y_i}{\sum_{i=1}^{N} \phi_i}$

Locally Weighted Projection Regression: Definition [(6), (1), (7)]

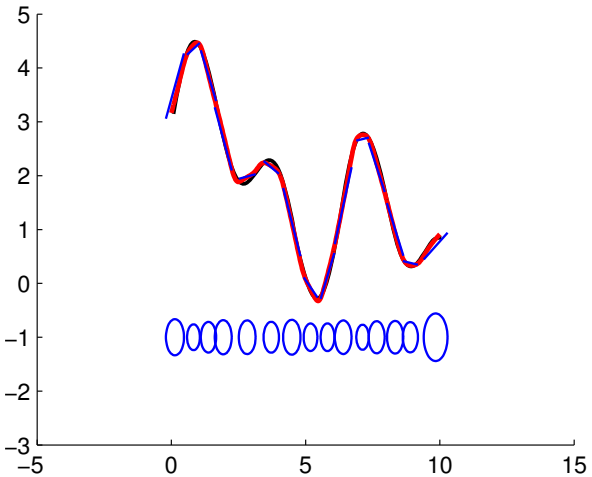
- Incremental algorithm which approximates non-linear functions with huge input space taking into account only relevant information
- LWPR is based on Locally Weighted Regression and Partial Least Squares



- $\phi_i = e^{-\frac{1}{2}(x-c_i)D_i(x-c_i)}$
- $y_i = \beta_i x$
- $\hat{y} = \frac{\sum_{i=1}^{N} \phi_i y_i}{\sum_{i=1}^{N} \phi_i}$

Locally Weighted Projection Regression: Definition [(6), (1), (7)]

- Incremental algorithm which approximates non-linear functions with huge input space taking into account only relevant information
- LWPR is based on Locally Weighted Regression and Partial Least Squares

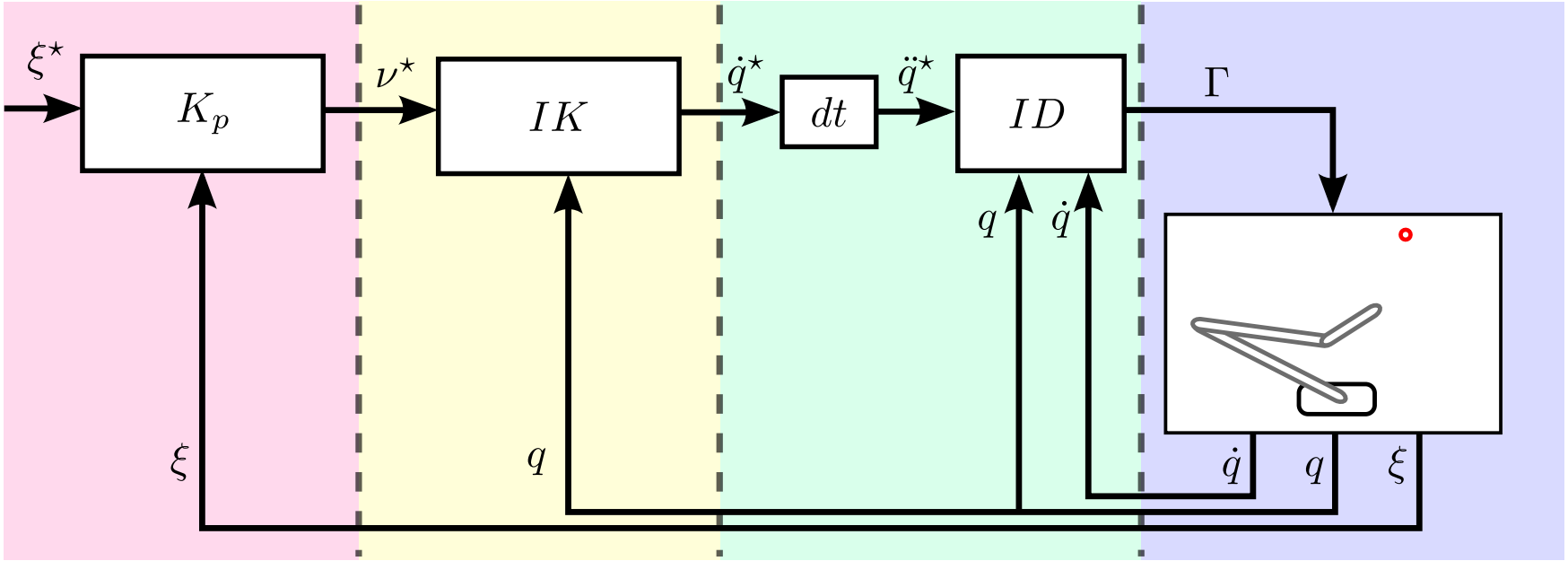


- $\phi_i = e^{-\frac{1}{2}(x-c_i)D_i(x-c_i)}$
- $y_i = \beta_i x$
- $\hat{y} = \frac{\sum_{i=1}^{N} \phi_i y_i}{\sum_{i=1}^{N} \phi_i}$

LWPR supply the derivative of the learnt model (5).

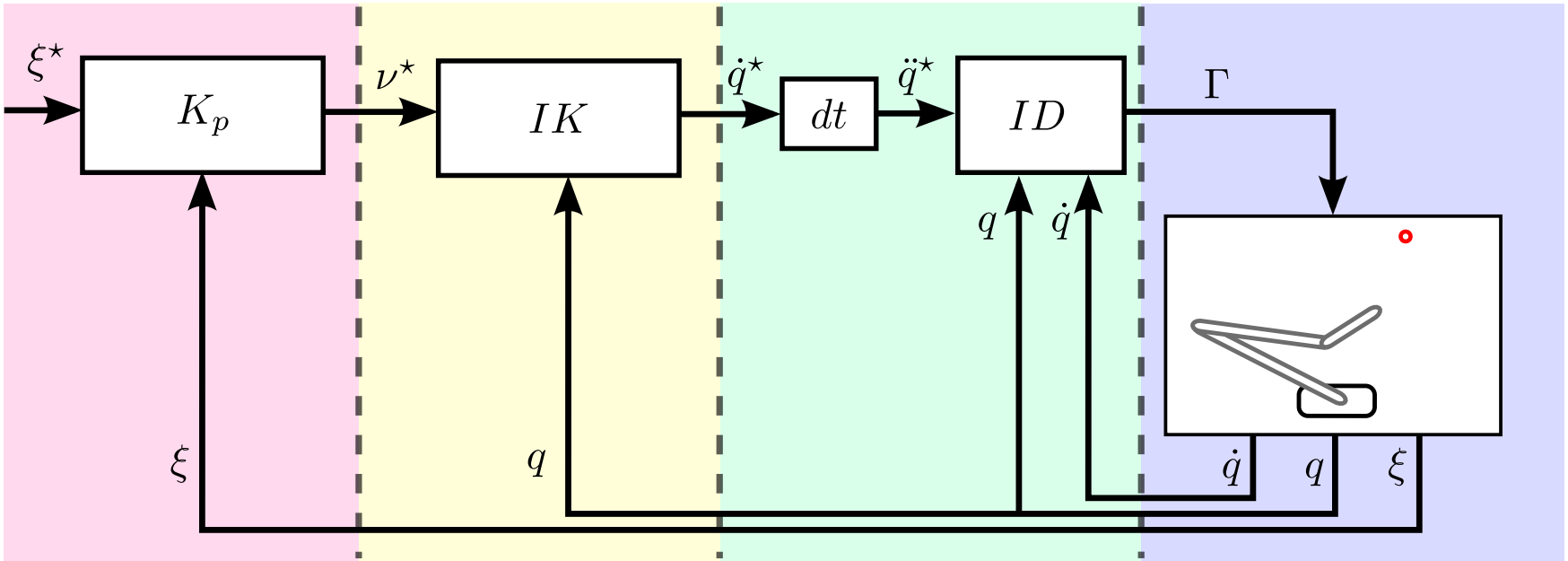
$$\frac{\partial \hat{y}}{\partial x} = \frac{1}{\Phi} \sum_{i=1}^{N} \left(\frac{\partial \phi_i}{\partial x} y_i + \phi_i \frac{\partial y_i}{\partial x} \right) - \frac{1}{\Phi^2} \sum_{i=1}^{N} \phi_i y_i \sum_{k=1}^{N} \frac{\partial \phi_k}{\partial x}$$

Our approach



Our claim: At the velocity level, one should rather learn forward mappings ((8)) for each task because ...

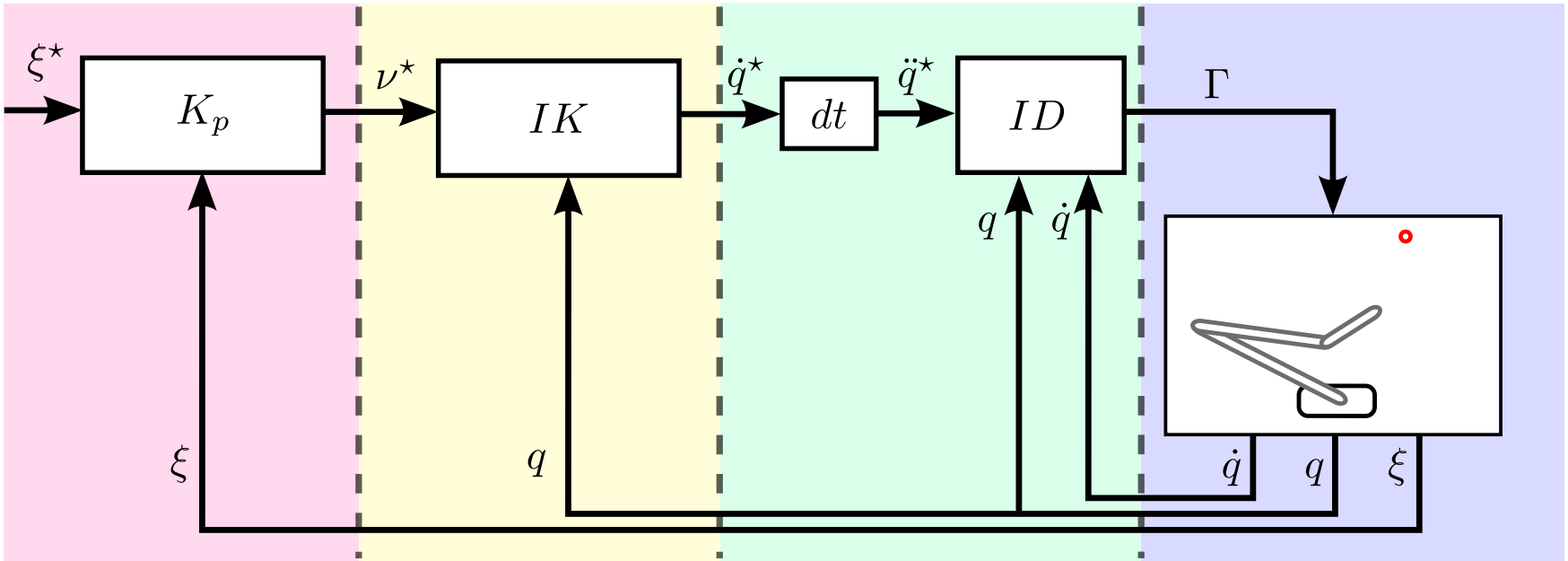
Our approach



Our claim: At the velocity level, one should rather learn forward mappings ((8)) for each task because ...

- ... directly learning inverse mappings leads to a loss of information about the redundant nature of the system;

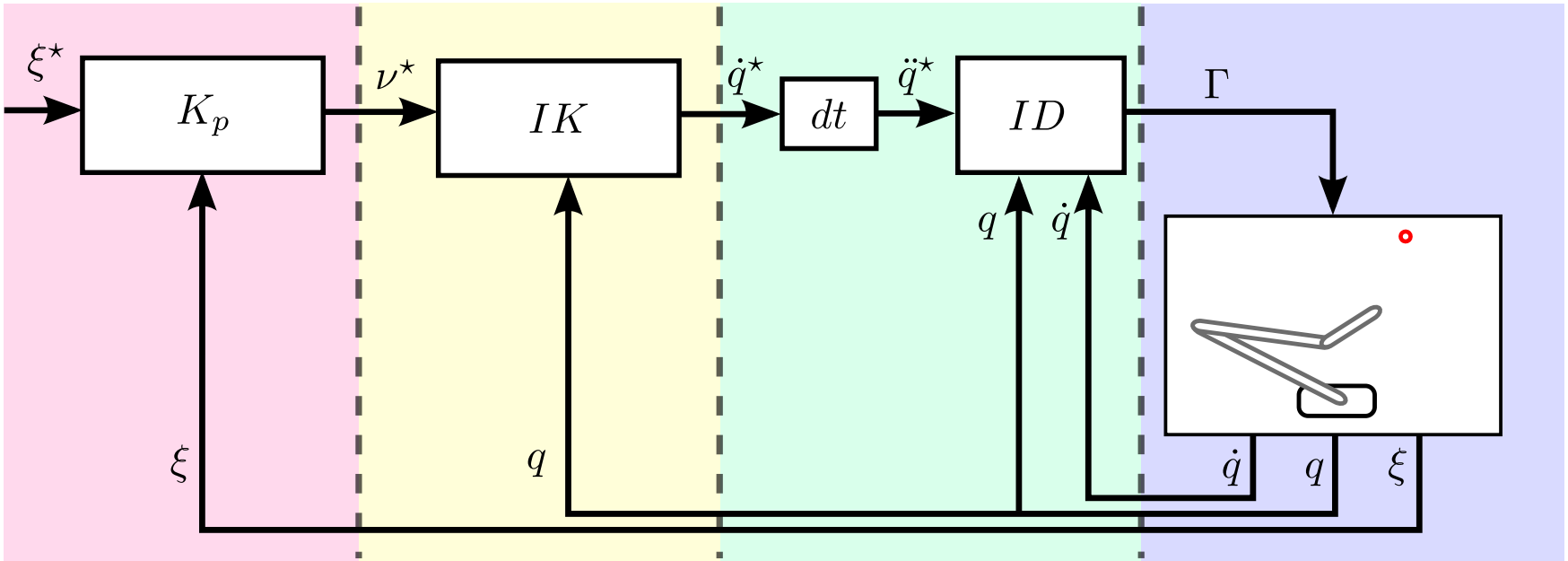
Our approach



Our claim: At the velocity level, one should rather learn forward mappings ((8)) for each task because ...

- ... directly learning inverse mappings leads to a loss of information about the redundant nature of the system;
- ... *a priori* task combination requires to (re)learn everything again when modifying the task combination.

Our approach

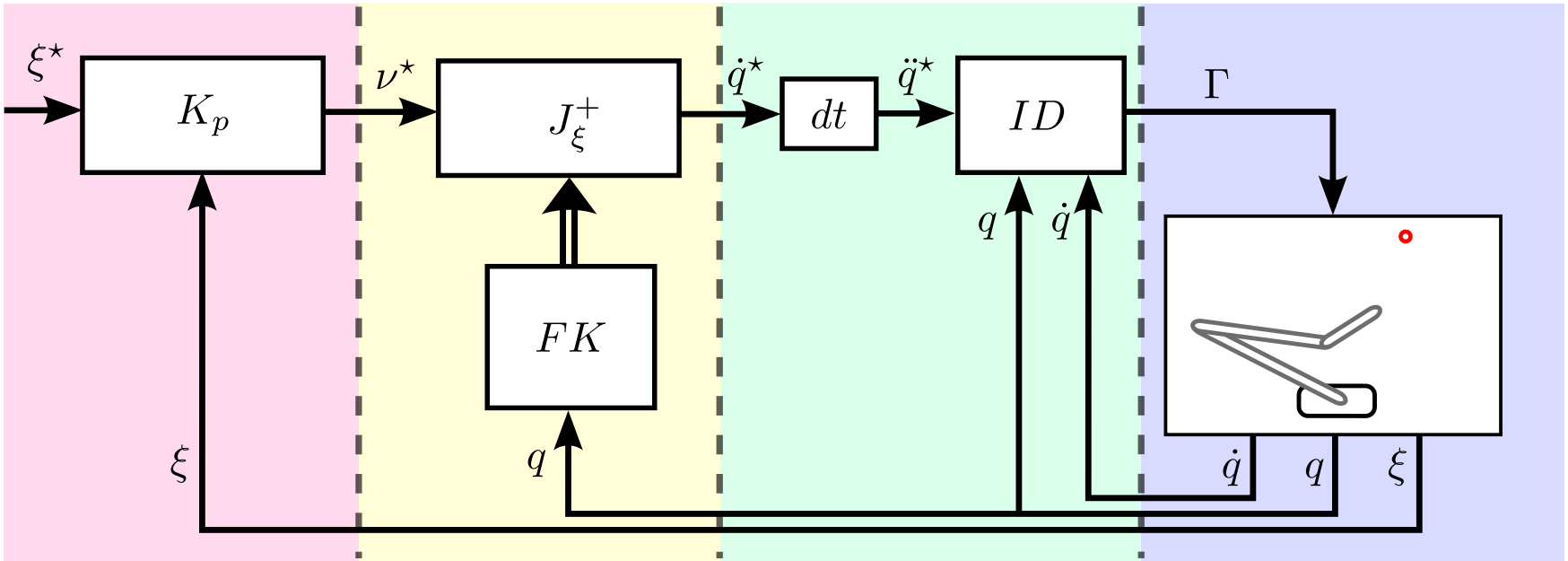


Our claim: At the velocity level, one should rather learn forward mappings ((8)) for each task because ...

- ... directly learning inverse mappings leads to a loss of information about the redundant nature of the system;
- ... *a priori* task combination requires to (re)learn everything again when modifying the task combination.

So, to control redundancy, we have chosen to:

Our approach



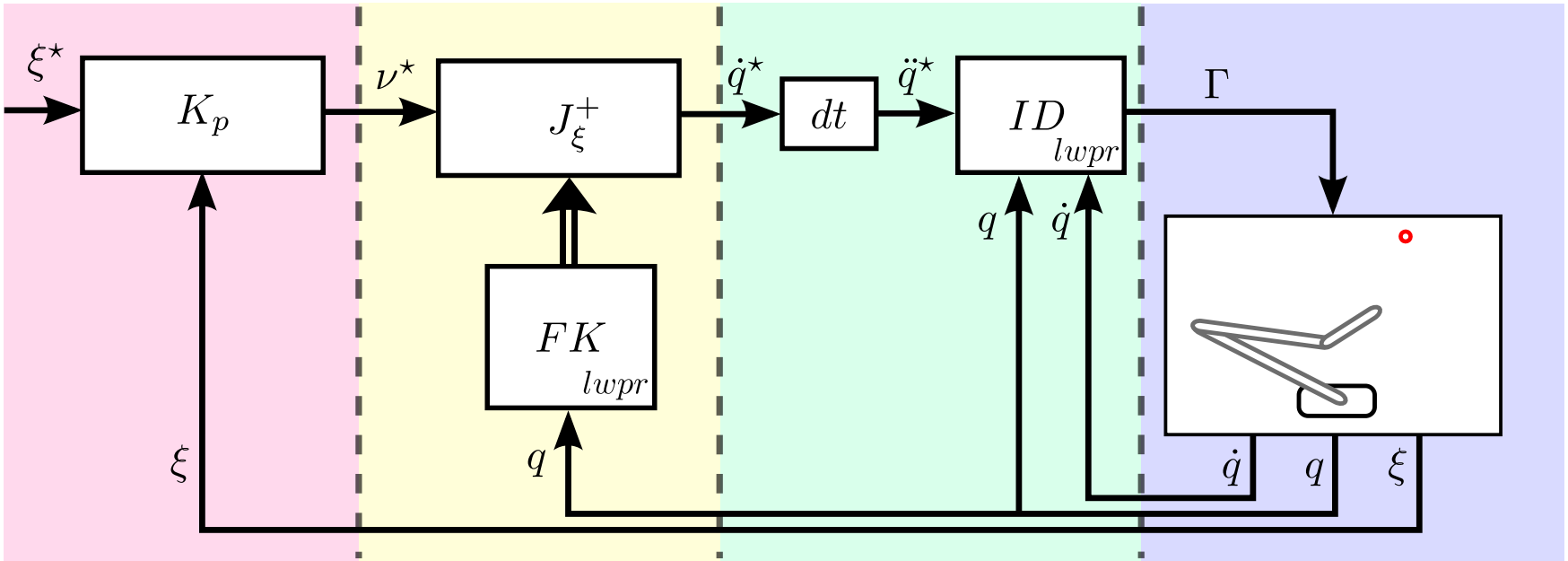
Our claim: At the velocity level, one should rather learn forward mappings ((8)) for each task because ...

- ... directly learning inverse mappings leads to a loss of information about the redundant nature of the system;
- ... *a priori* task combination requires to (re)learn everything again when modifying the task combination.

So, to control redundancy, we have chosen to:

- learn separately velocity kinematics and dynamics models;

Our approach



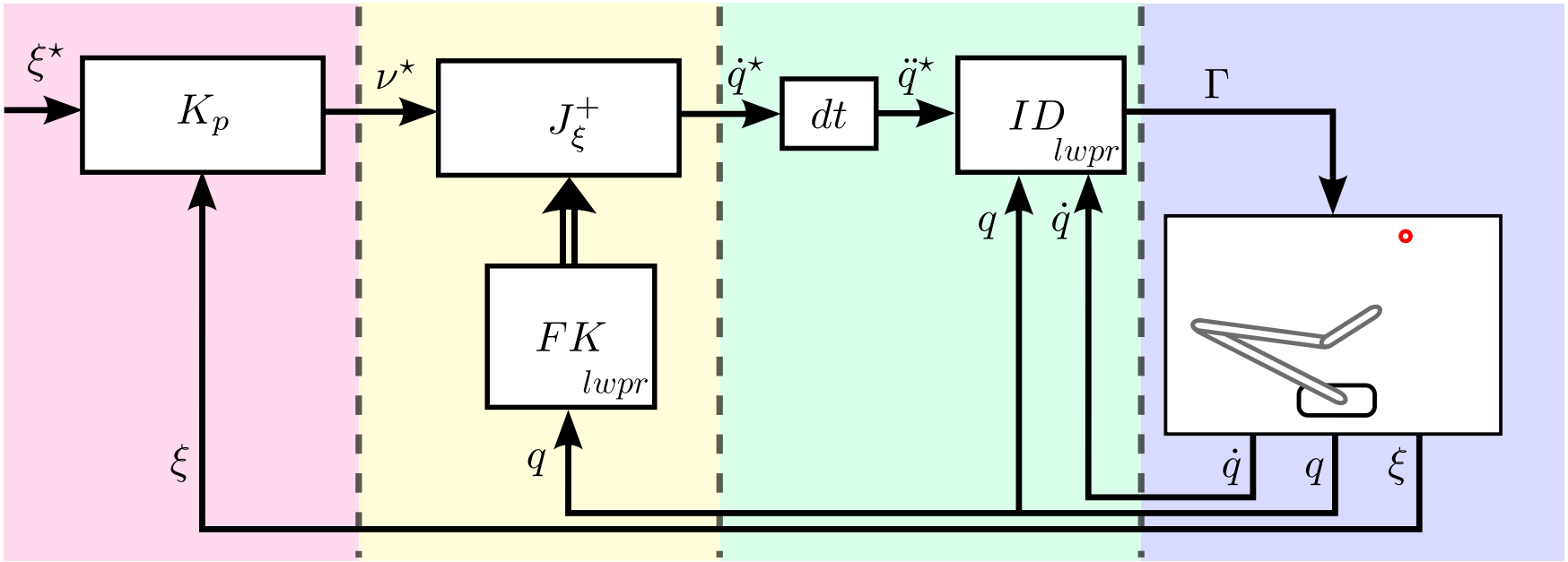
Our claim: At the velocity level, one should rather learn forward mappings ((8)) for each task because ...

- ... directly learning inverse mappings leads to a loss of information about the redundant nature of the system;
- ... *a priori* task combination requires to (re)learn everything again when modifying the task combination.

So, to control redundancy, we have chosen to:

- learn separately velocity kinematics and dynamics models;

Our approach



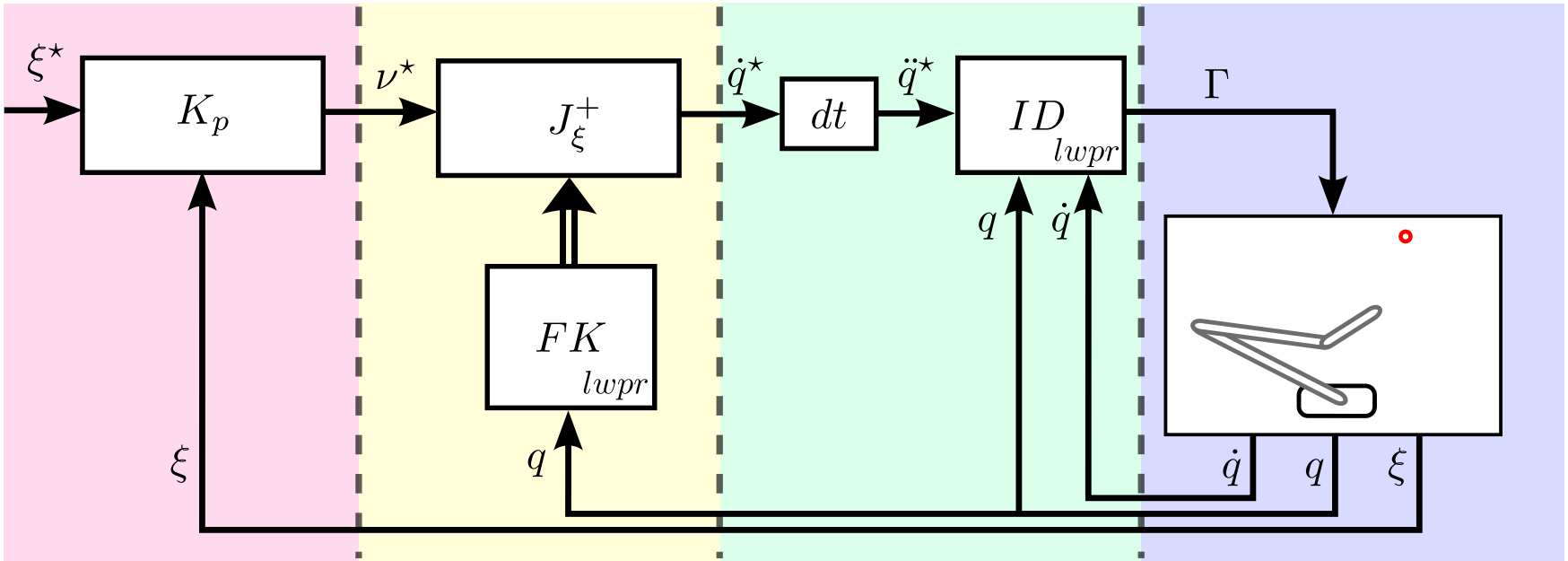
Our claim: At the velocity level, one should rather learn forward mappings ((8)) for each task because ...

- ... directly learning inverse mappings leads to a loss of information about the redundant nature of the system;
- ... *a priori* task combination requires to (re)learn everything again when modifying the task combination.

So, to control redundancy, we have chosen to:

- learn separately velocity kinematics and dynamics models;
- learn models on the whole state space;

Our approach



Our claim: At the velocity level, one should rather learn forward mappings ((8)) for each task because ...

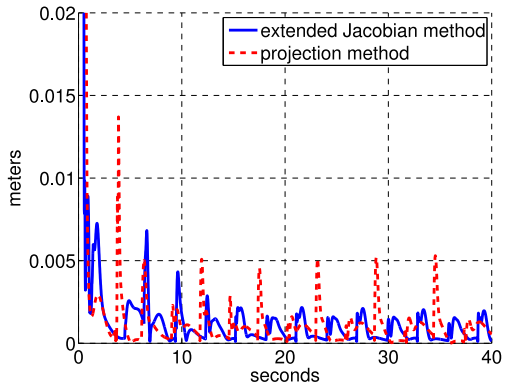
- ... directly learning inverse mappings leads to a loss of information about the redundant nature of the system;
- ... *a priori* task combination requires to (re)learn everything again when modifying the task combination.

So, to control redundancy, we have chosen to:

- learn separately velocity kinematics and dynamics models;
- learn models on the whole state space;
- learn forward velocity kinematics and invert it analytically.

Figure: Point to point task with limited joint space babbling: 2000 samples.

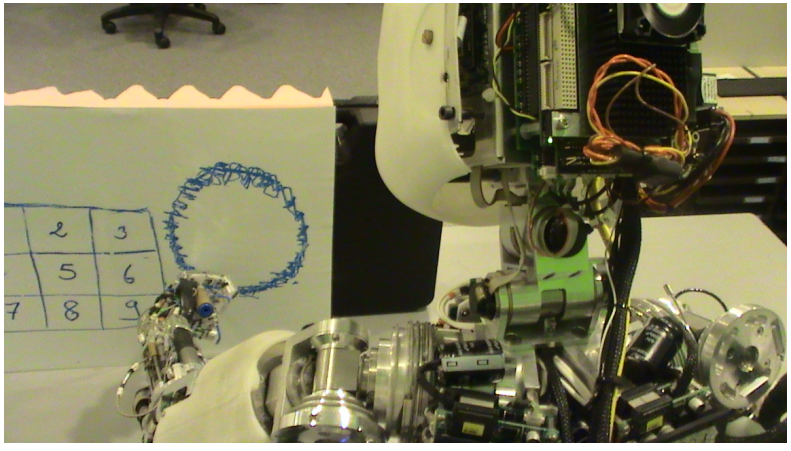
Controlling redundancy with LWPR

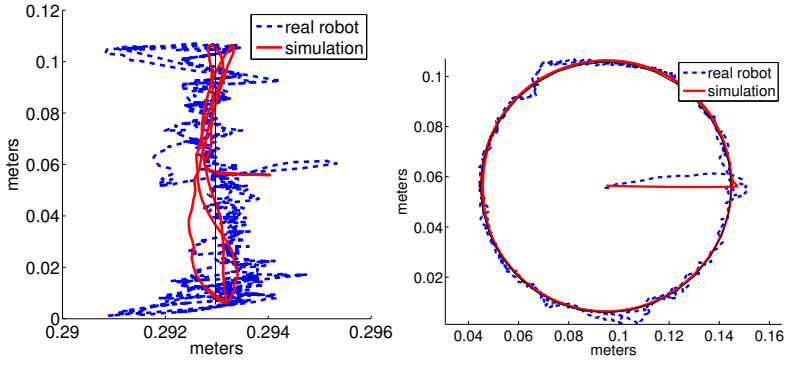


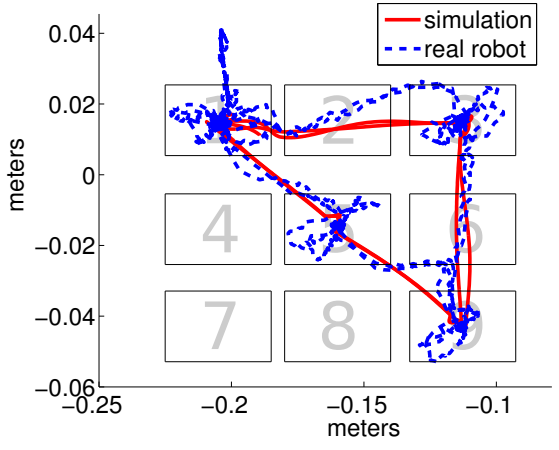
compatible task incompatible task normalized task space error

Figure: Point to point task with adaptation to an unknown external force (after 2 seconds)

Learning velocity kinematics on the real robot







Summary

Summary

- Combination of model-based control techniques and incremental learning algorithms.

Summary

- Combination of model-based control techniques and incremental learning algorithms.
- From the simulated world to iCub.

Summary

- Combination of model-based control techniques and incremental learning algorithms.
- From the simulated world to iCub.
- Dynamics control and learning requires torque control.

Summary

- Combination of model-based control techniques and incremental learning algorithms.
- From the simulated world to iCub.
- Dynamics control and learning requires torque control.

$\hookrightarrow$ looking for the next iCub technological improvements.

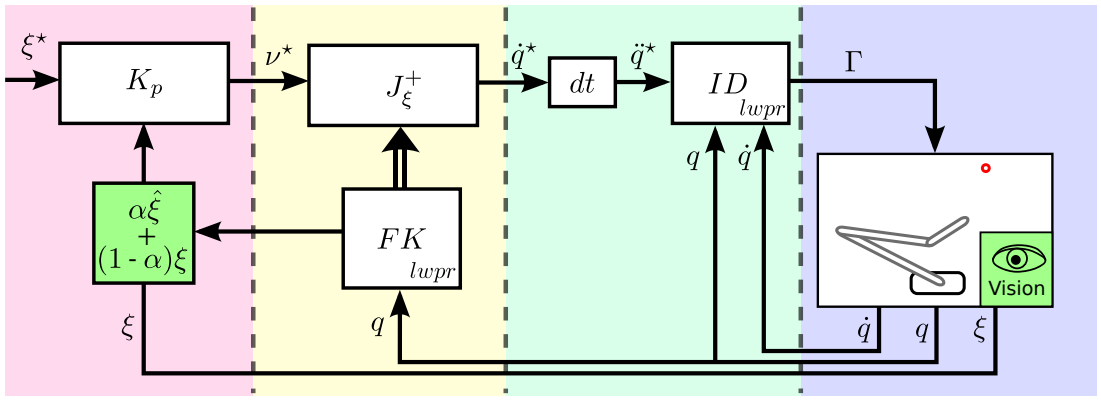
Summary

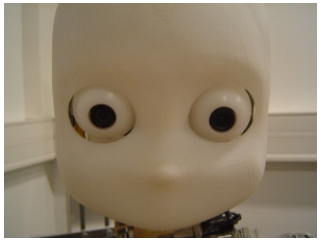
- Combination of model-based control techniques and incremental learning algorithms.
- From the simulated world to iCub.
- Dynamics control and learning requires torque control.

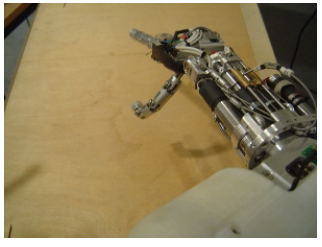
$\hookrightarrow$ looking for the next iCub technological improvements.

- ⋆ Clercq, C. and Salaun, C. and Padois, V. and Sigaud, O. (2010). On the Limitations of a Model Learning Approach for a velocity Controlled Humanoid Robot. Submitted to IEEE RAM.
- ⋆ Salaun, C. and Padois, V. and Sigaud, O. (2010). Learning Forward Models for the Operational Space Control of Redundant Robots. From motor to interaction learning in robots Springer, publisher. Pages 169-192.
- ⋆ Salaun, C. and Padois, V. and Sigaud, O. (2009). A Two-Level Model of Anticipation-Based Motor Learning for Whole Body Motion. LNAI 5499: Anticipatory Behavior in Adaptive Learning Systems: From Psychological Theories to Artificial Cognitive Systems Springer, publisher. Pages 229–246.
- ⋆ Salaun, C. and Padois, V. and Sigaud, O. (2009). Control of Redundant Robots Using Learned Models: An Operational Space Control Approach. Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems. Pages 878–885. Saint-Louis, USA.

Close the loop at the vision level and combine proprioceptive and exteroceptive state estimation.

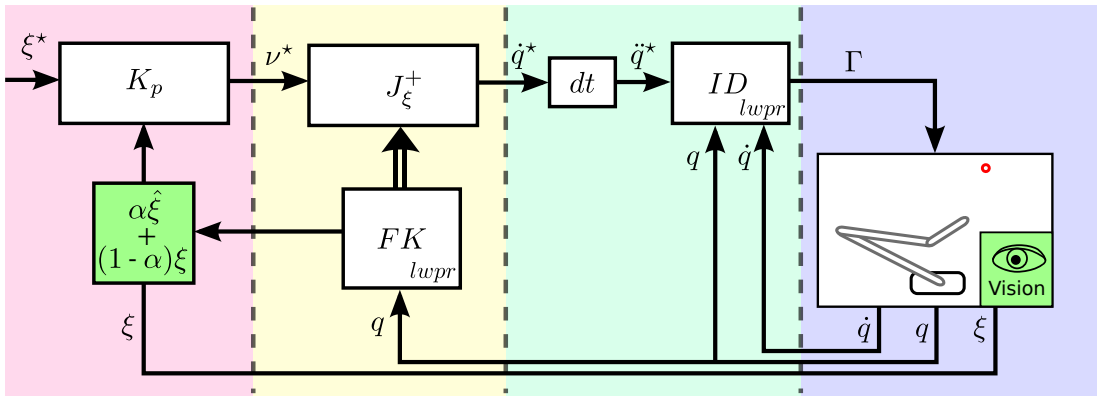


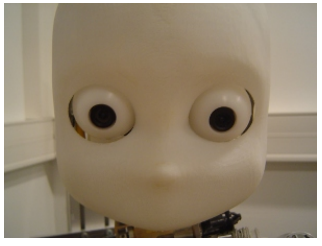


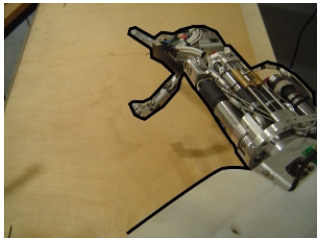


- Locate end-effectors;
- Differentiate objects, humans, the robot limbs;
- Multi-sensors state estimation.

Close the loop at the vision level and combine proprioceptive and exteroceptive state estimation.

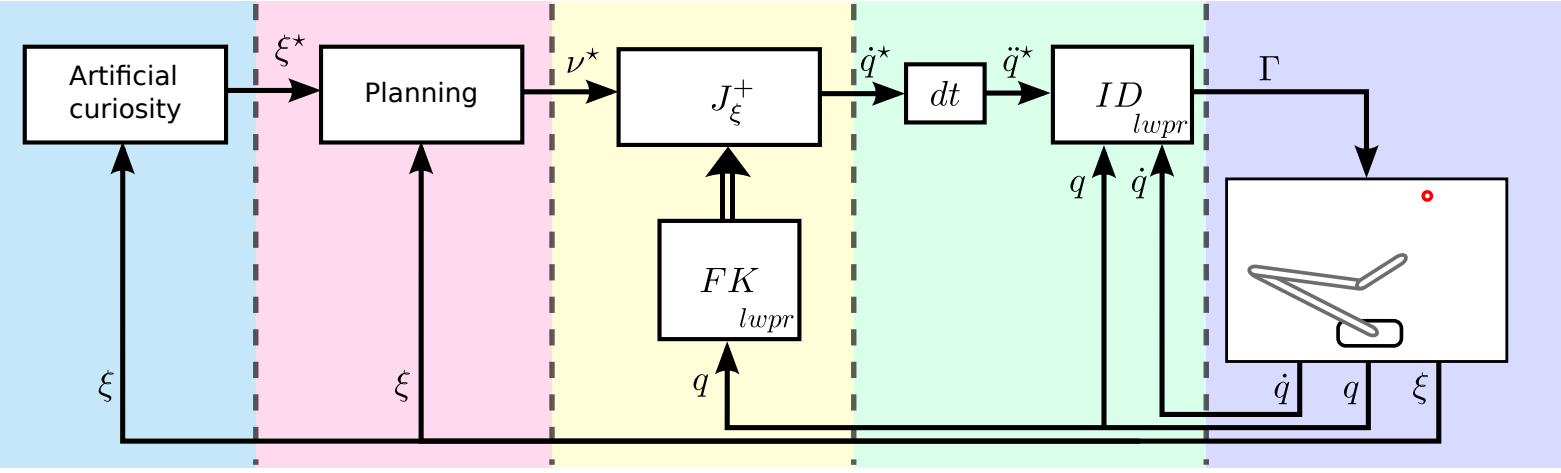






- Locate end-effectors;
- Differentiate objects, humans, the robot limbs;
- Multi-sensors state estimation.

Solve the exloration problem.

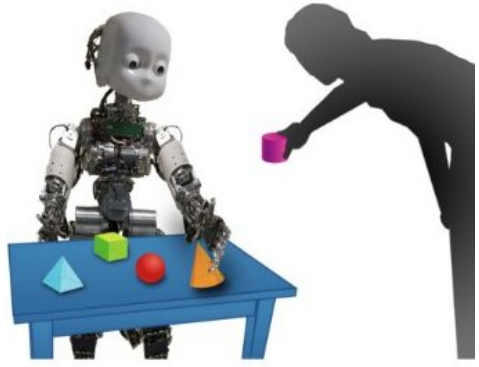


- Close the loop at a higher level;
- Combine artificial curiosity and human guidance.

Acknowledgments

- RobotCub for:
  - providing us with a great robot;
  - allowing both national and international scientific collaboration;
  - constant technical support.
- My colleagues and students working on this project:
  - Olivier Sigaud;
  - Pierre-Yves Oudeyer & Olivier Ly (INRIA Flowers);
  - David Filliat (ENSTA);
  - Jean-Christophe Bailli (Gostai);
  - Camille Salaün (PhD student);
  - Charles Clercq (former CS Master Student, now at IIT);
  - Guillaume Sicard (CS Master student);
  - Pierre Griffault, Kevin Hoang and Abhishek Agarwal (former interns).

Thank you for your attention.



References

[1] A. D'Souza, S. Vijayakumar, and S. Schaal, "Learning inverse kinematics," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, 2001, pp. 298–303.

[2] S. Schaal, C. G. Atkeson, and S. Vijayakumar, "Scalable techniques from nonparametric statistics for real time robot learning," *Applied Intelligence*, vol. 17, no. 1, pp. 49–60, 2002.

[3] J. Peters and S. Schaal, "Learning to control in operational space," *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 197–212, 2008.

[4] D. Mitrovic, S. Klanke, and S. Vijayakumar, "Adaptive Optimal Control for Redundantly Actuated Arms," in *From Animals to Animats 10: 10th International Conference on Simulation of Adaptive Behavior, Sab 2008, Osaka, Japan, July 7-12, 2008, Proceedings*. Springer, 2008, p. 93.

[5] S. Klanke, S. Vijayakumar, and S. Schaal, "A library for locally weighted projection regression," *The Journal of Machine Learning Research*, vol. 9, pp. 623–626, 2008.

[6] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally weighted learning," *Artificial Intelligence Review*, vol. 11, no. 1, pp. 11–73, 1997.

[7] S. Vijayakumar, A. D'Souza, and S. Schaal, "LWPR: A scalable method for incremental online learning in high dimensions," Edinburgh University Press, Tech. Rep., 2005.

[8] L. Natale, F. Nori, G. Metta, and G. Sandini, "Learning precise 3d reaching in a humanoid robot," in *Proceedings of the IEEE International Conference of Development and Learning (ICDL)*, London, UK, July 2007, pp. 1–6.