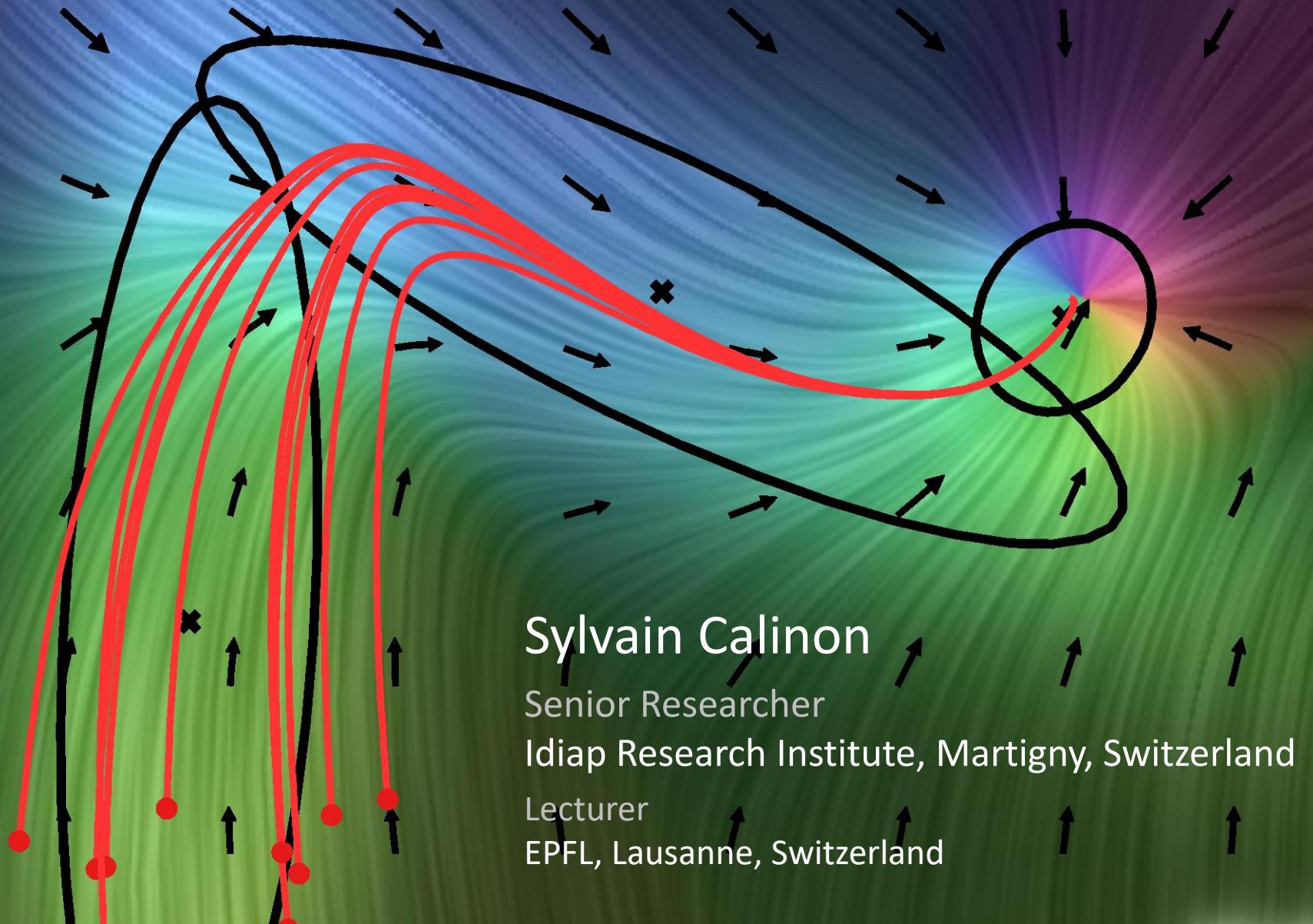


Motion primitives



Sylvain Calinon

Senior Researcher

Idiap Research Institute, Martigny, Switzerland

Lecturer

EPFL, Lausanne, Switzerland

Artificial Intelligence for Society

Research Groups:

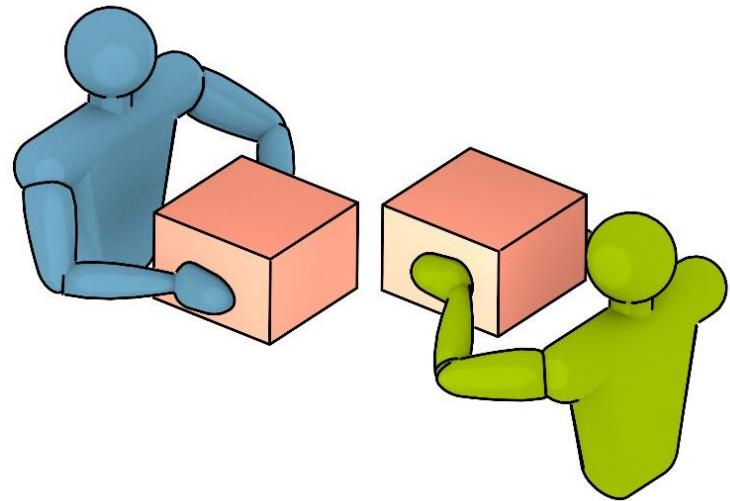
- Speech & Audio Processing
- Natural Language Understanding
- Perception & Activity Understanding
- Machine Learning
- Social Computing
- Biometrics Security and Privacy
- Biosignal Processing
- Computational Bioimaging
- Energy Informatics
- Uncertainty Quantification and Optimal Design
- **Robot Learning & Interaction**



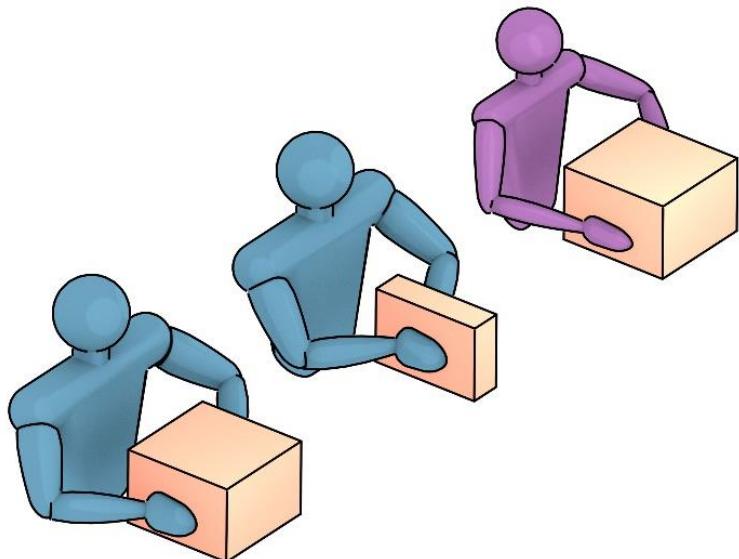
Research
Education
Technology transfer



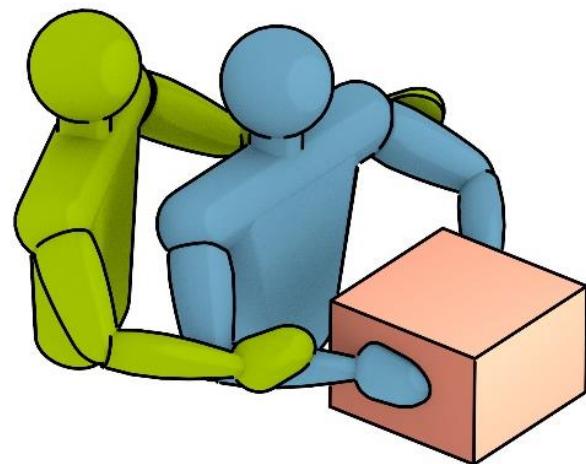
Learning from demonstration



Observational learning

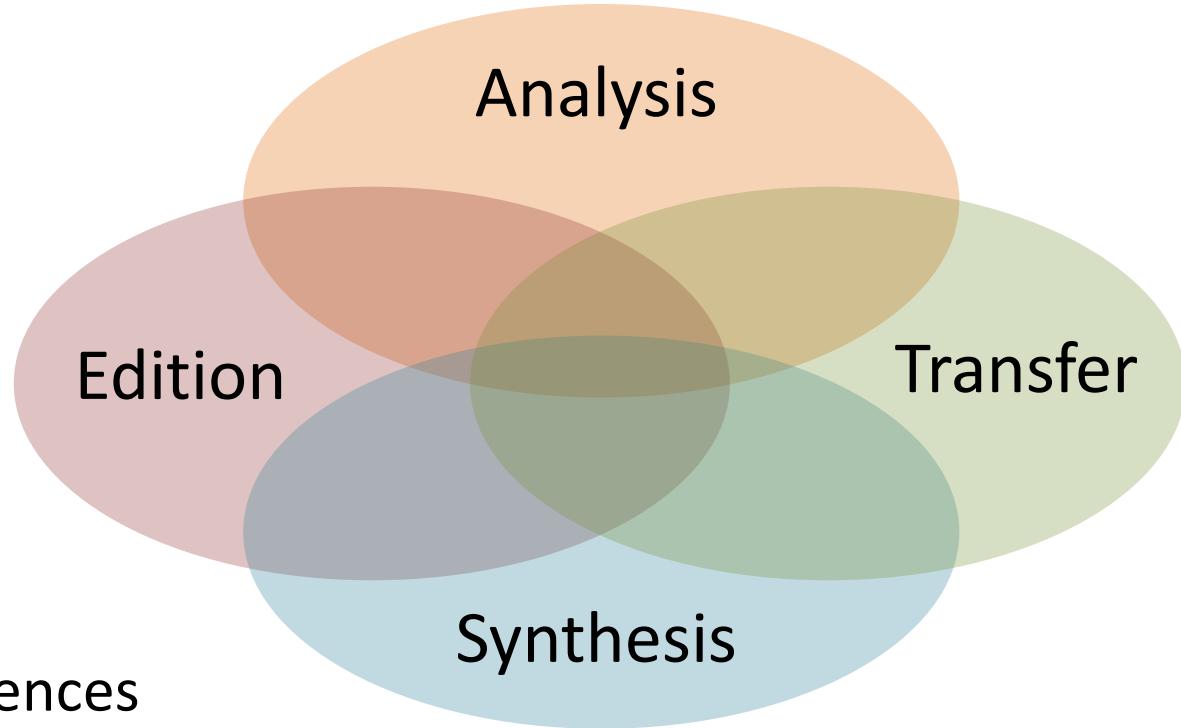


Correspondence problems



Kinesthetic teaching

Statistical, geometrical and dynamical representations of movements



- Robotics
- Human movement sciences
- Sensorimotor control
- Neurosciences
- Biomechanics
- Computer graphics
- Sport sciences
- ...

Outline

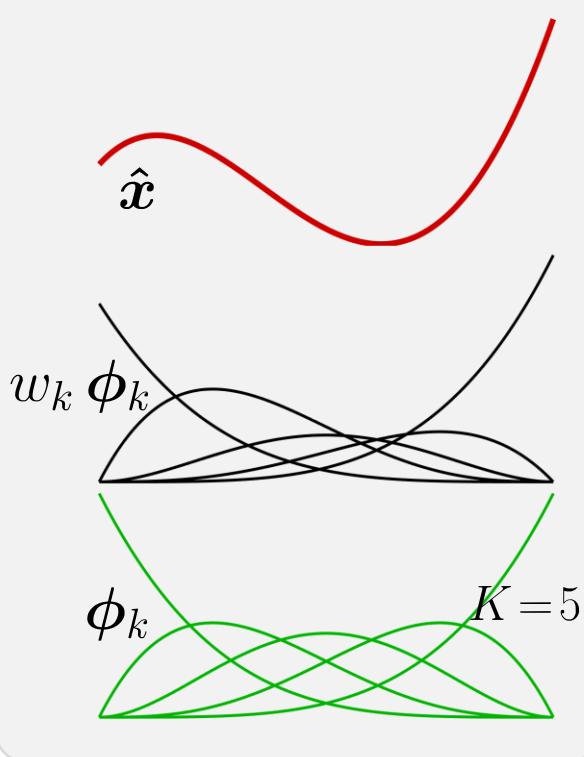
- **Superposition with basis functions**
 - Bezier curves
 - Locally weighted regression (LWR)
 - Gaussian mixture regression (GMR)
 - Fourier series for periodic motion and ergodic control
- **Dynamical movement primitives (DMP)**
 - Probabilistic movement primitives (ProMP)
- **Superposition Vs fusion**
 - Product of Gaussians
- **Model predictive control (MPC)**
 - Linear quadratic tracking (LQT)
 - Task-parameterized movement models
- **Differential geometry**
 - Riemannian manifolds

Superposition with basis functions

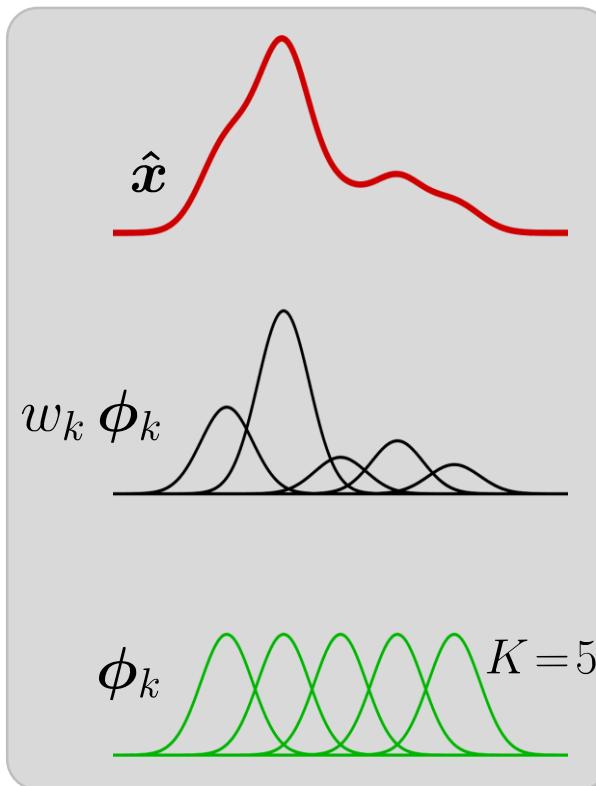
$$\hat{x} = \sum_{k=1}^K w_k \phi_k$$



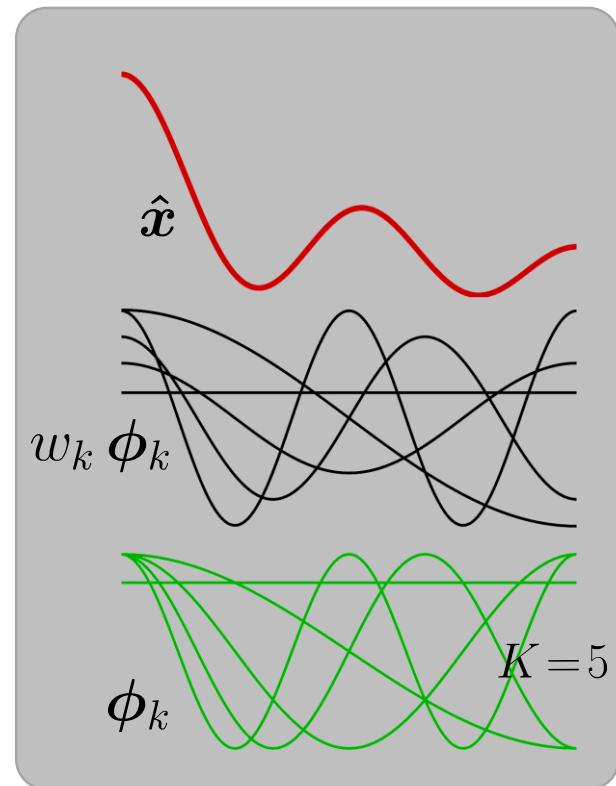
Bézier curves
(Bernstein bases)



**Locally weighted
regression (radial bases)**



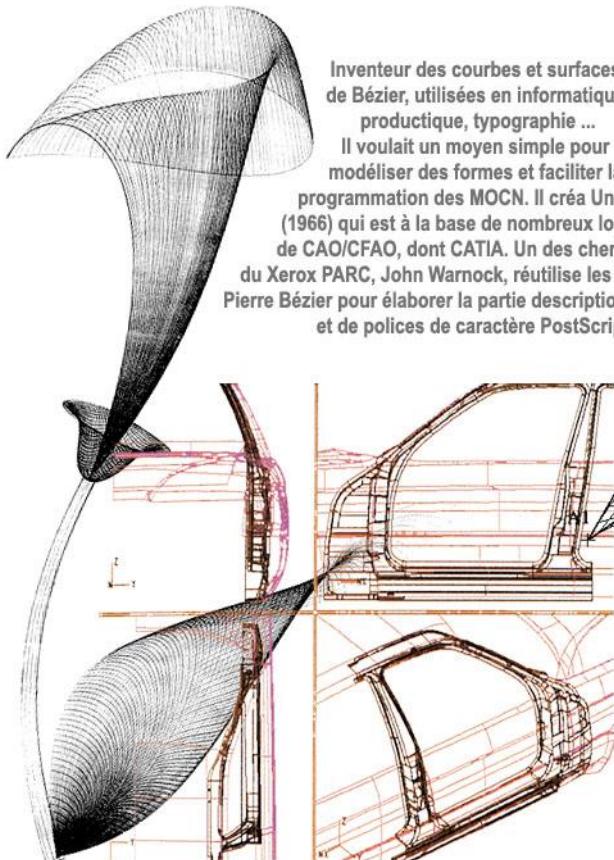
Fourier series
(cosine bases)



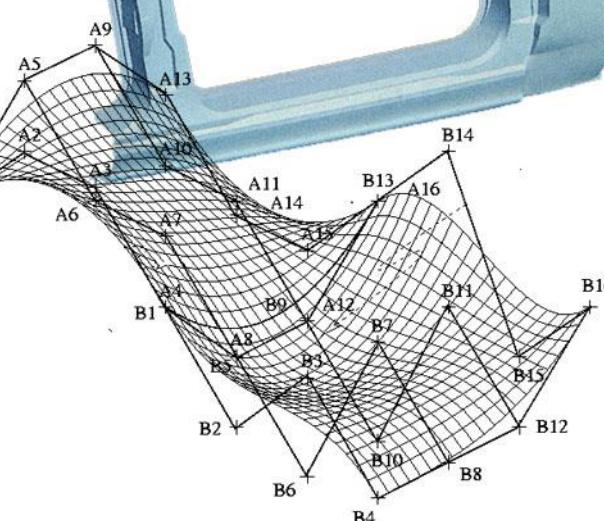
Bezier curves

Pierre Bézier

01/09/1910 - 25/11/1999



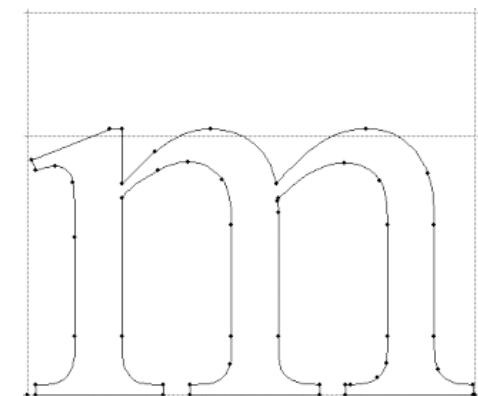
Inventeur des courbes et surfaces de Bézier, utilisées en informatique productique, typographie ... Il voulait un moyen simple pour modéliser des formes et faciliter la programmation des MOCN. Il créa Unisurf (1966) qui est à la base de nombreux logiciels de CAO/CFAO, dont CATIA. Un des chercheurs du Xerox PARC, John Warnock, réutilise les travaux de Pierre Bézier pour élaborer la partie description de courbes et de polices de caractère PostScript.



Sergei Bernstein



Paul de Casteljau



Bezier curves

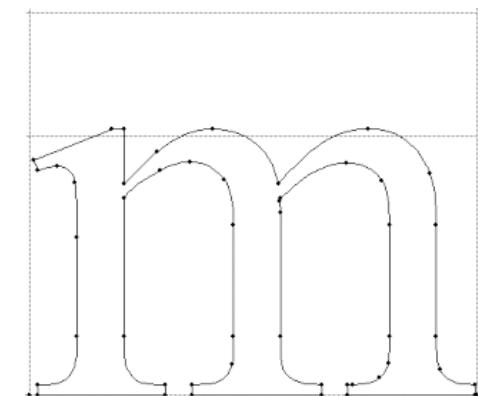
- Mathematical basis for Bézier curves: **Bernstein polynomials**, known since 1912 but applied to graphics only 50 years later.
- **Pierre Bézier** patented and popularized, but did not invent the Bézier curves and Bézier surfaces that are now used in most computer-aided design and computer graphics systems.
- The study of these curves was first developed in 1959 by **Paul de Casteljau** at Citroën, where he was initially not permitted to publish his work. Pierre Bézier used them to design cars at Renault.
- Bézier curves are often used to define **3D paths** as well as 2D curves for **keyframe interpolation**.
- Bézier curves are now very frequently used to edit movements and animations.
- **TrueType fonts** use quadratic Bézier curves, PostScript and SVG use cubic Bézier curves.



Sergei Bernstein

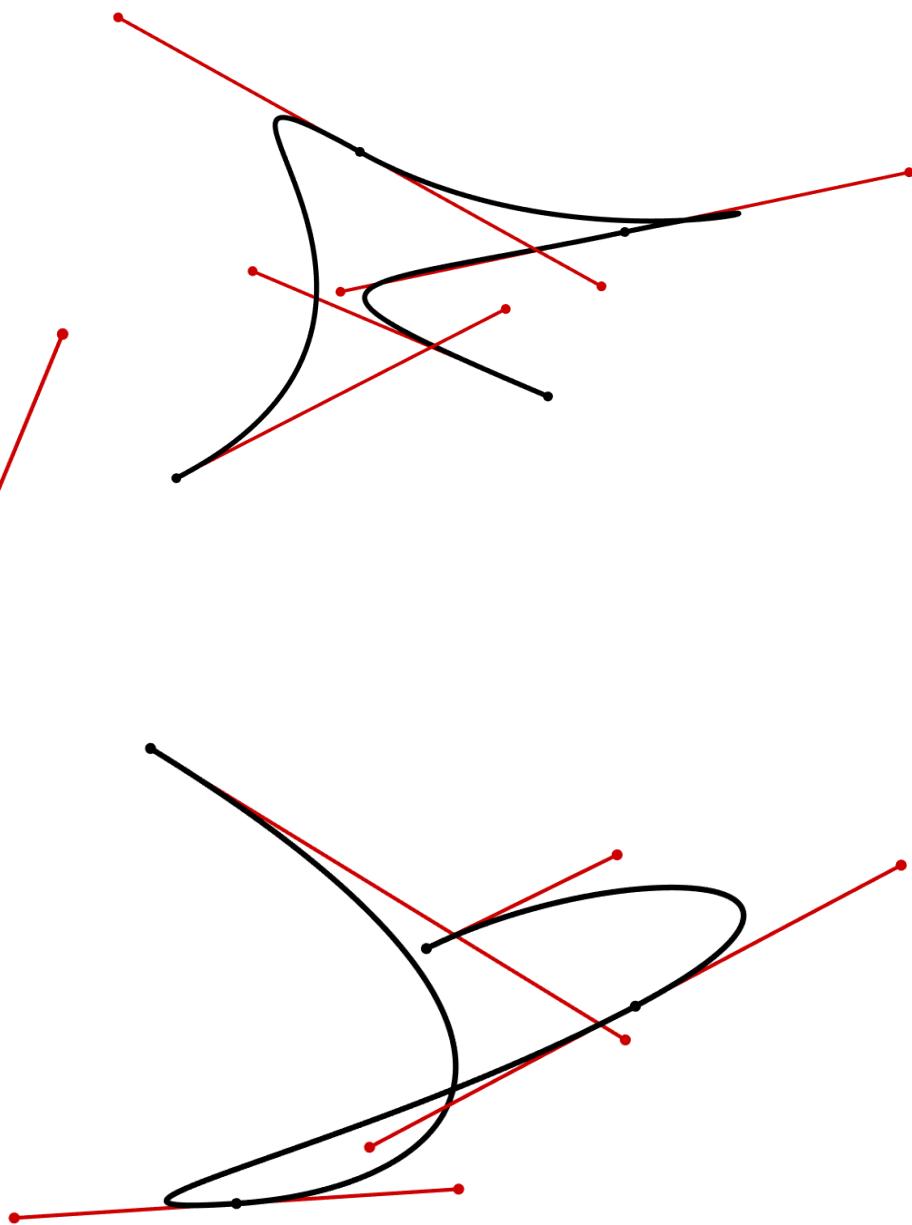
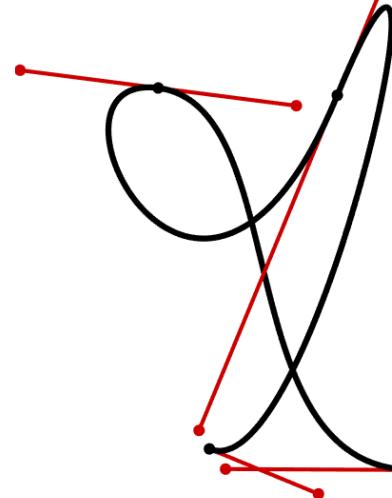
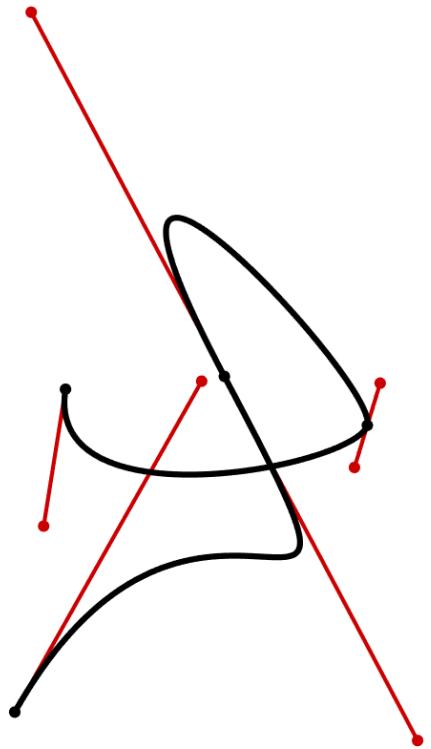


Paul de Casteljau



Bezier curves

As the curve is completely contained in the convex hull of its control points, the points can be graphically displayed and used to manipulate the curve intuitively.



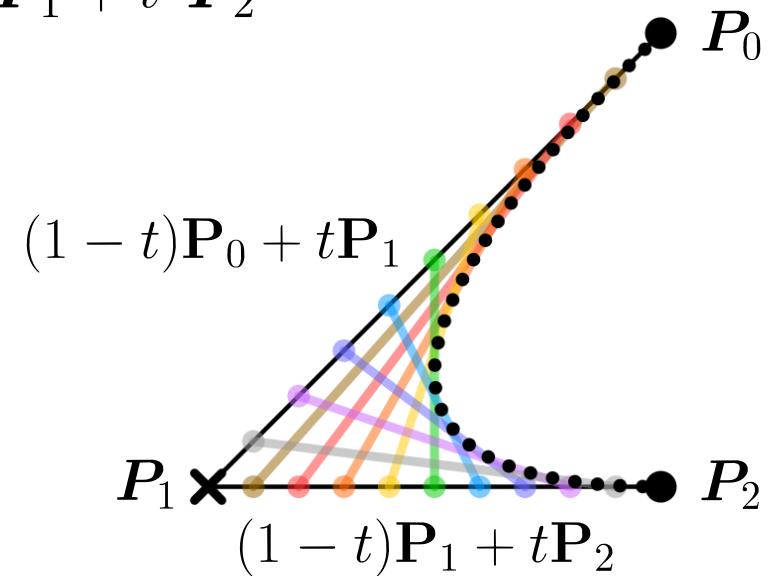
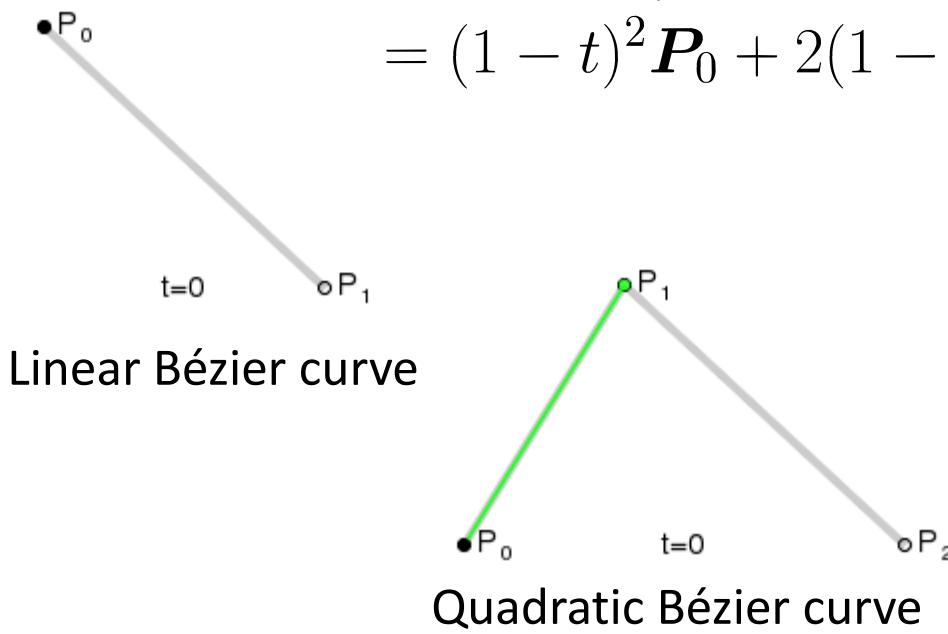
Quadratic Bézier curves

For $0 \leq t \leq 1$, a **linear Bézier curve** is the line traced by

$$\mathbf{B}_{P_0, P_1}(t) = (1 - t)\mathbf{P}_0 + t\mathbf{P}_1$$

For $0 \leq t \leq 1$, a **quadratic Bézier curve** is the path traced by

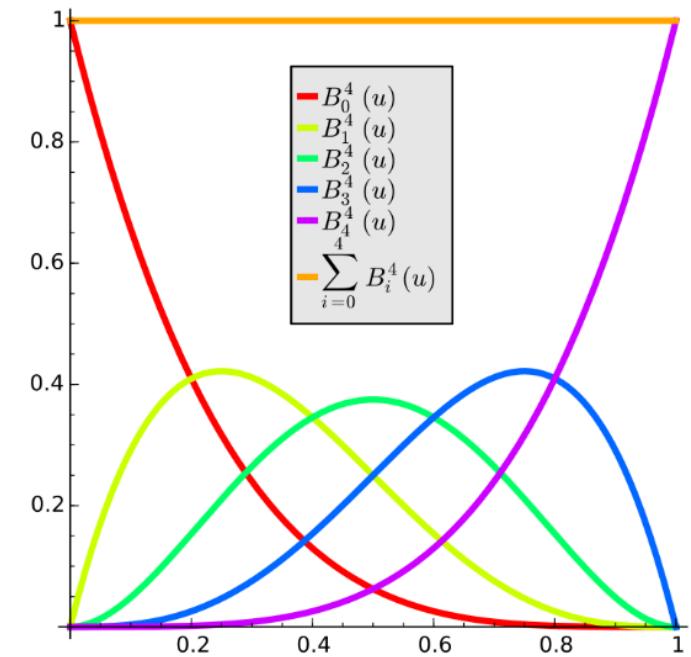
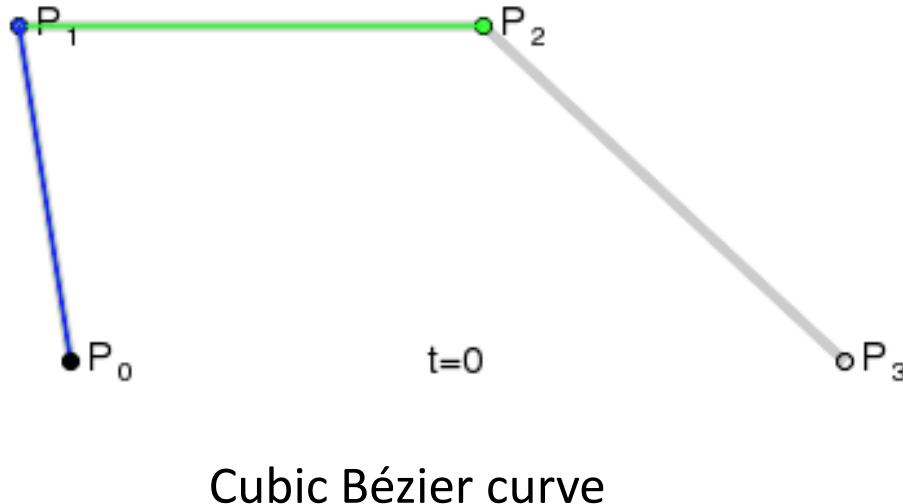
$$\begin{aligned}\mathbf{B}_{P_0, P_1, P_2}(t) &= (1 - t)\mathbf{B}_{P_0, P_1}(t) + t\mathbf{B}_{P_1, P_2}(t) \\ &= (1 - t)((1 - t)\mathbf{P}_0 + t\mathbf{P}_1) + t((1 - t)\mathbf{P}_1 + t\mathbf{P}_2) \\ &= (1 - t)^2\mathbf{P}_0 + 2(1 - t)t\mathbf{P}_1 + t^2\mathbf{P}_2\end{aligned}$$



Cubic Bézier curves

For $0 \leq t \leq 1$, a **cubic Bézier curve** is the path traced by

$$\begin{aligned}\mathbf{B}_{P_0, P_1, P_2, P_3}(t) &= (1-t) \mathbf{B}_{P_0, P_1, P_2}(t) + t \mathbf{B}_{P_1, P_2, P_3}(t) \\ &= (1-t)^3 \mathbf{P}_0 + 3(1-t)^2 t \mathbf{P}_1 + 3(1-t)t^2 \mathbf{P}_2 + t^3 \mathbf{P}_3\end{aligned}$$



Bezier curves of degree n

$$\hat{\mathbf{x}} = \sum_{k=1}^K w_k \phi_k$$



For $0 \leq t \leq 1$, a recursive definition for the **Bézier curve of degree n** can be expressed as a linear interpolation of a pair of corresponding points in two Bézier curves of degree $n - 1$, namely

$$\mathbf{B}(t) = \sum_{i=0}^n b_{i,n}(t) \mathbf{P}_i$$

with

$$b_{i,n}(t) = \binom{n}{i} (1-t)^{n-i} t^i$$

the **Bernstein polynomials** of degree n,

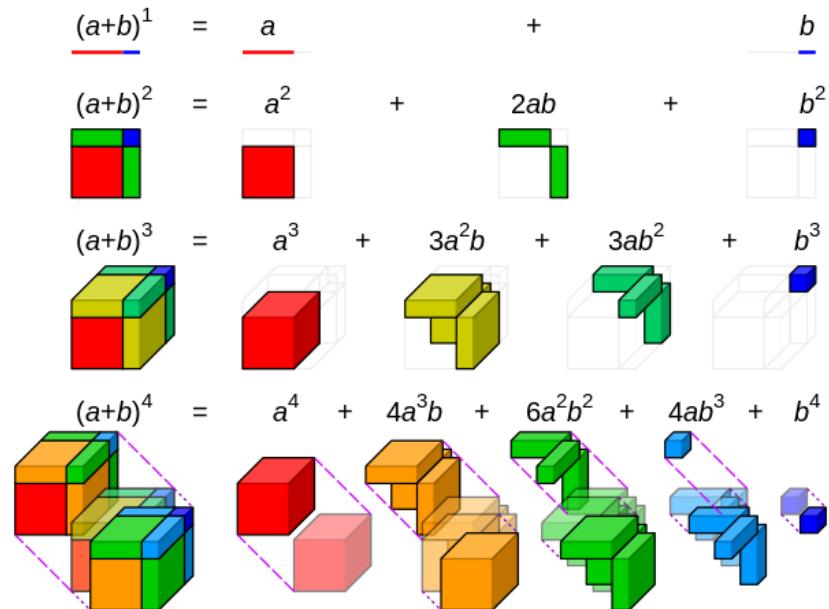
where $\binom{n}{i} = \frac{n!}{i!(n-i)!}$ are **binomial coefficients**.

Binomial coefficients

$$B(t) = \sum_{i=0}^n \frac{n!}{i!(n-i)!} (1-t)^{n-i} t^i P_i$$

$$\begin{array}{ccccccccc} & & & & 1 & & & & \\ & & & & 1 & 1 & 1 & & \\ & & & & 1 & 2 & 1 & & \\ & & & & 1 & 3 & 3 & 1 & \\ & & & & 1 & 4 & 6 & 4 & 1 \\ & & & & 1 & 5 & 10 & 10 & 5 & 1 \end{array}$$

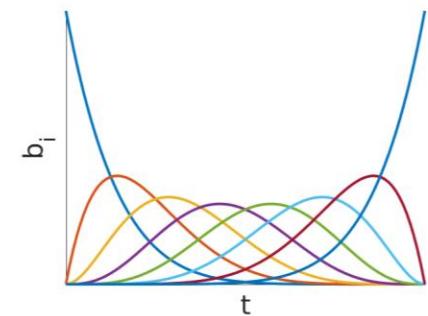
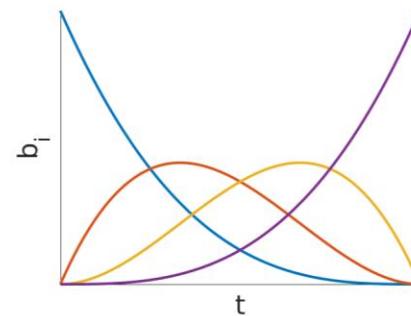
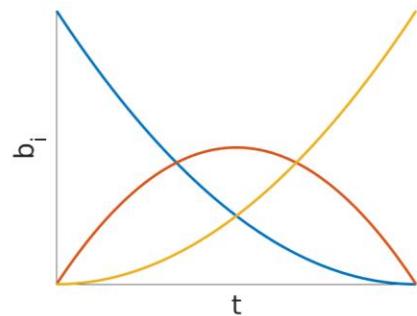
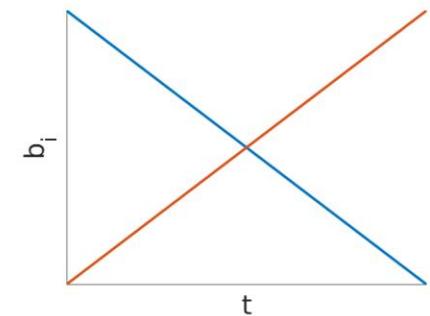
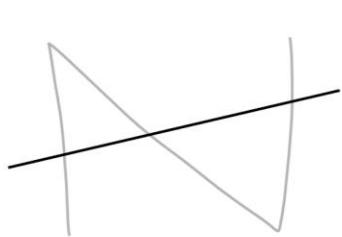
$$\begin{aligned} B(t) &= \boxed{t^0 = 1} \quad (1-t)^4 \quad P_0 \\ &+ 4t \quad (1-t)^3 \quad P_1 \\ &+ 6t^2 \quad (1-t)^2 \quad P_2 \\ &+ 4t^3 \quad (1-t) \quad P_3 \\ &+ t^4 \quad \quad \quad P_4 \\ &\quad \quad \quad (1-t)^0 = 1 \end{aligned}$$



$$\begin{aligned} B(t) &= \boxed{(1-t)^5 \quad P_0} \\ &+ 5t \quad (1-t)^4 \quad P_1 \\ &+ 10t^2 \quad (1-t)^3 \quad P_2 \\ &+ 10t^3 \quad (1-t)^2 \quad P_3 \\ &+ 5t^4 \quad (1-t) \quad P_4 \\ &+ t^5 \quad P_5 \end{aligned}$$

Bezier curves - Summary

$$\hat{x} = \sum_{k=1}^K w_k \phi_k$$



$$B(t) = \sum_{i=0}^n \binom{n}{i} (1-t)^{n-i} t^i P_i$$

Outline

- **Superposition with basis functions**

- Bezier curves

- Locally weighted regression (LWR)**

- Gaussian mixture regression (GMR)

- Fourier series for periodic motion and ergodic control

- **Dynamical movement primitives (DMP)**

- Probabilistic movement primitives (ProMP)

- **Superposition Vs fusion**

- Product of Gaussians

- **Model predictive control (MPC)**

- Linear quadratic tracking (LQT)

- Task-parameterized movement models

- **Differential geometry**

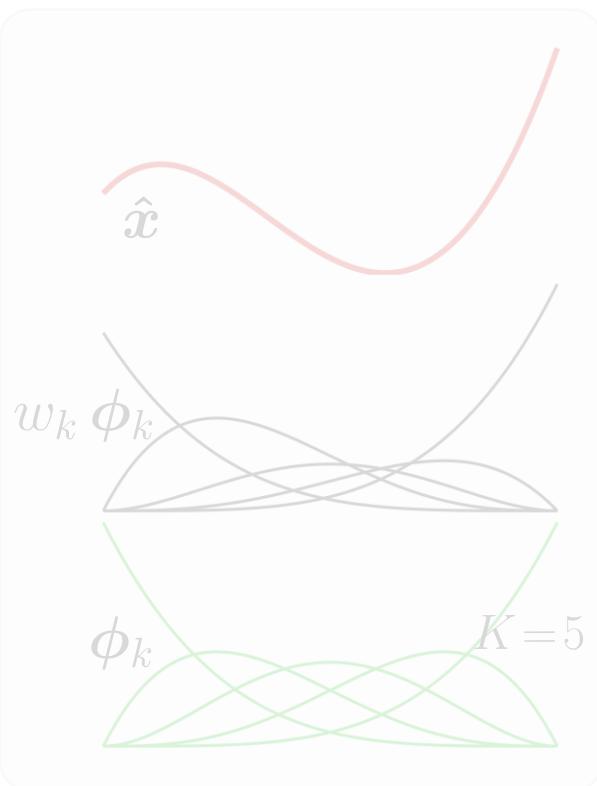
- Riemannian manifolds

Superposition with basis functions

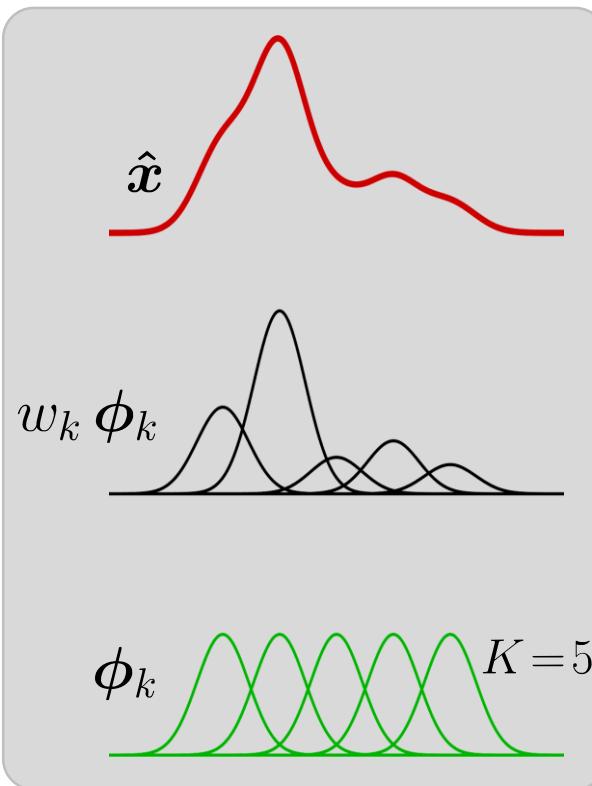
$$\hat{x} = \sum_{k=1}^K w_k \phi_k$$



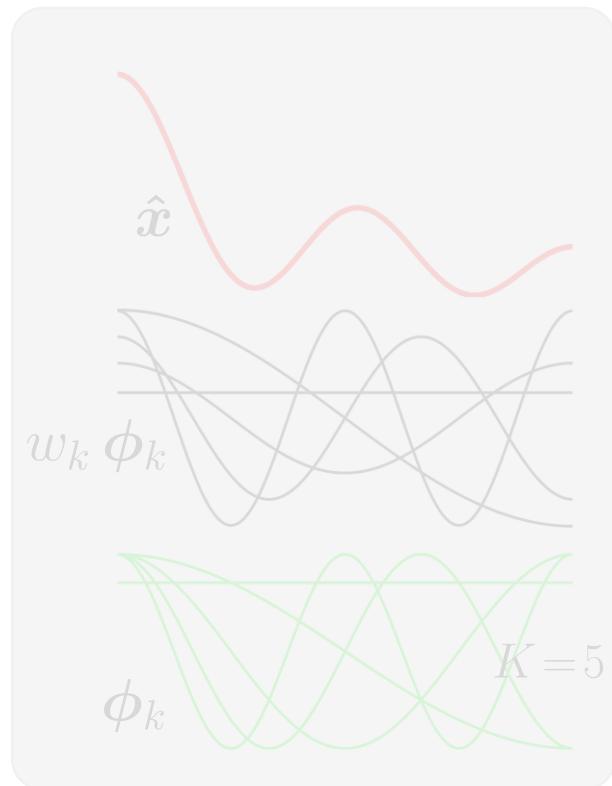
Bézier curves
(Bernstein bases)



Locally weighted
regression (radial bases)



Fourier series
(cosine bases)



Multivariate ordinary least squares

By describing the input data as $\mathbf{X} \in \mathbb{R}^{N \times D^I}$ and the output data as $\mathbf{Y} \in \mathbb{R}^{N \times D^O}$, we want to find $\mathbf{A} \in \mathbb{R}^{D^I \times D^O}$ such that $\mathbf{Y} = \mathbf{X}\mathbf{A}$.

A solution can be found by minimizing the L_2 -norm of the residuals

$$\begin{aligned}\hat{\mathbf{A}} &= \arg \min_{\mathbf{A}} \|\mathbf{Y} - \mathbf{X}\mathbf{A}\|_2^2 \\ &= \arg \min_{\mathbf{A}} (\mathbf{Y} - \mathbf{X}\mathbf{A})^\top (\mathbf{Y} - \mathbf{X}\mathbf{A}) \\ &= \arg \min_{\mathbf{A}} (\mathbf{Y}^\top \mathbf{Y} - 2\mathbf{A}^\top \mathbf{X}^\top \mathbf{Y} + \mathbf{A}^\top \mathbf{X}^\top \mathbf{X}\mathbf{A})\end{aligned}$$

Moore-Penrose
pseudoinverse

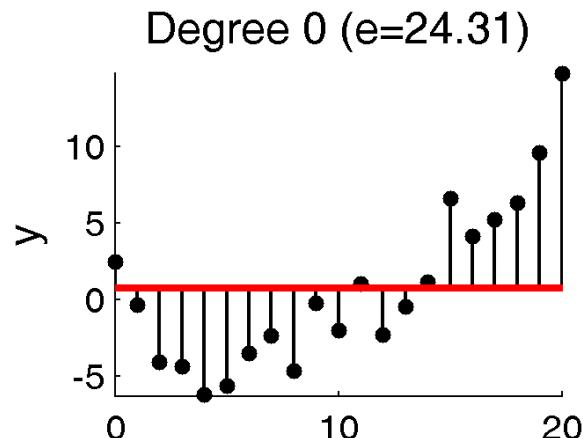
\mathbf{X}^\dagger

By differentiating with respect to \mathbf{A} and equating to zero

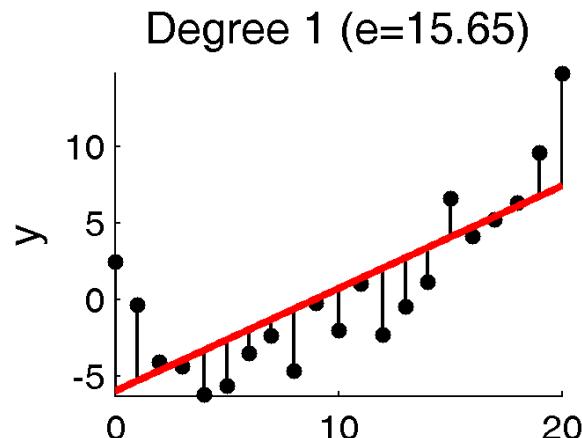
$$-2\mathbf{X}^\top \mathbf{Y} + 2\mathbf{X}^\top \mathbf{X}\mathbf{A} = \mathbf{0} \iff \hat{\mathbf{A}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}$$

Polynomial fitting with least squares

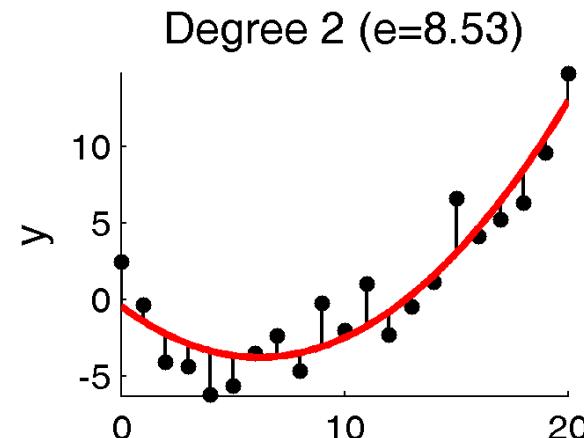
$$\hat{A} = X^\dagger Y$$



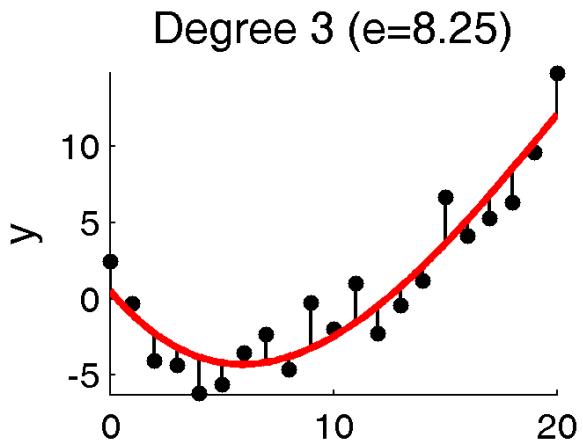
$$X = 1$$



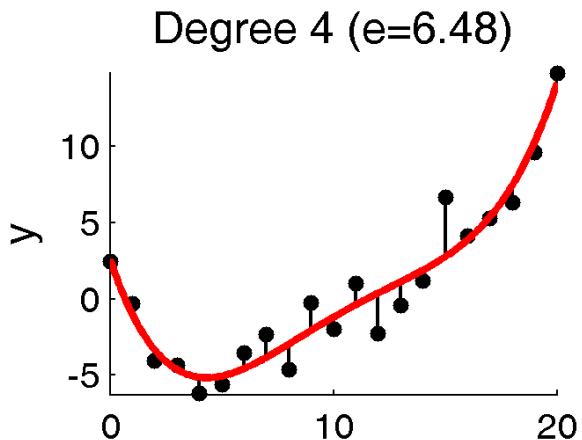
$$X = [x, 1]$$



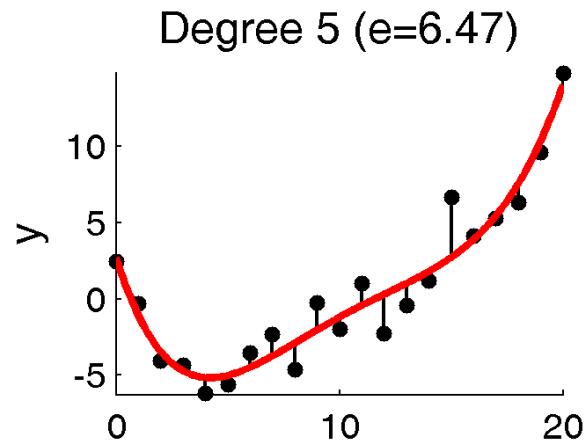
$$X = [x^2, x, 1]$$



$$X = [x^3, x^2, x, 1]$$



$$X = [x^4, x^3, \dots, 1]$$



$$X = [x^5, x^4, \dots, 1]$$

Weighted least squares

By describing the input data as $\mathbf{X} \in \mathbb{R}^{N \times D^I}$ and the output data as $\mathbf{Y} \in \mathbb{R}^{N \times D^O}$, we want to minimize

$$\begin{aligned}\hat{\mathbf{A}} &= \arg \min_{\mathbf{A}} \|\mathbf{Y} - \mathbf{X}\mathbf{A}\|_{2,\mathbf{W}}^2 \\ &= \arg \min_{\mathbf{A}} (\mathbf{Y} - \mathbf{X}\mathbf{A})^\top \mathbf{W} (\mathbf{Y} - \mathbf{X}\mathbf{A}) \\ &= \arg \min_{\mathbf{A}} (\mathbf{Y}^\top \mathbf{W} \mathbf{Y} - 2\mathbf{A}^\top \mathbf{X}^\top \mathbf{W} \mathbf{Y} + \mathbf{A}^\top \mathbf{X}^\top \mathbf{W} \mathbf{X} \mathbf{A})\end{aligned}$$

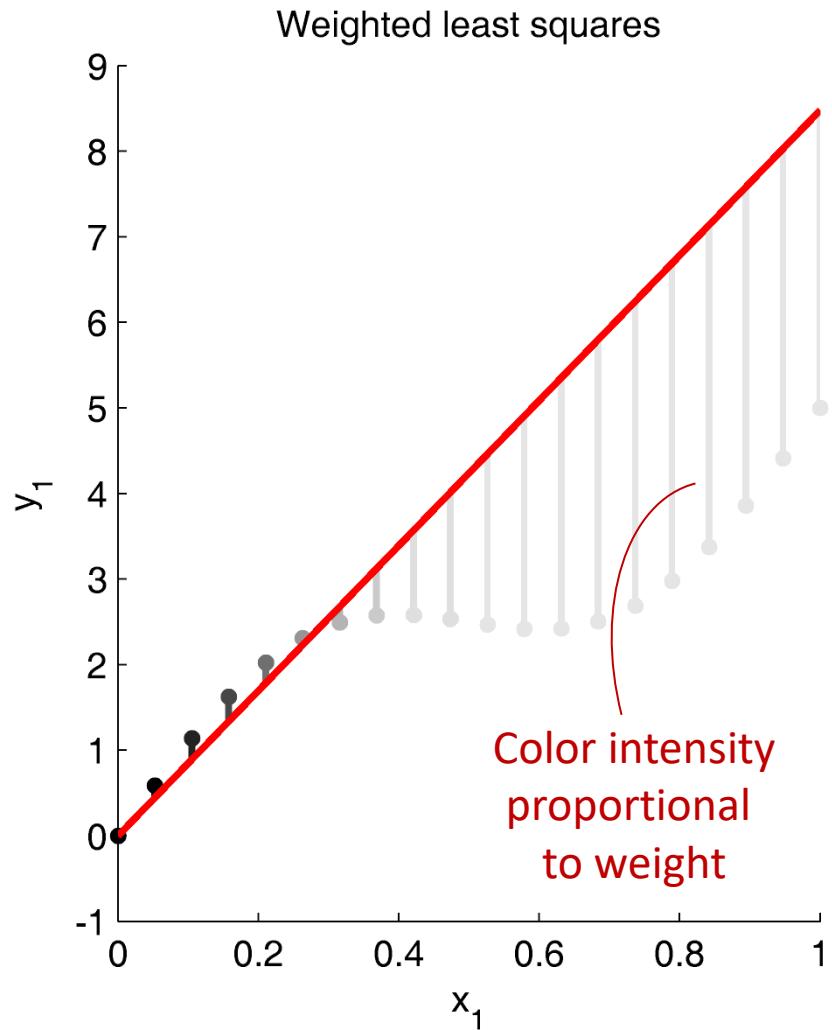
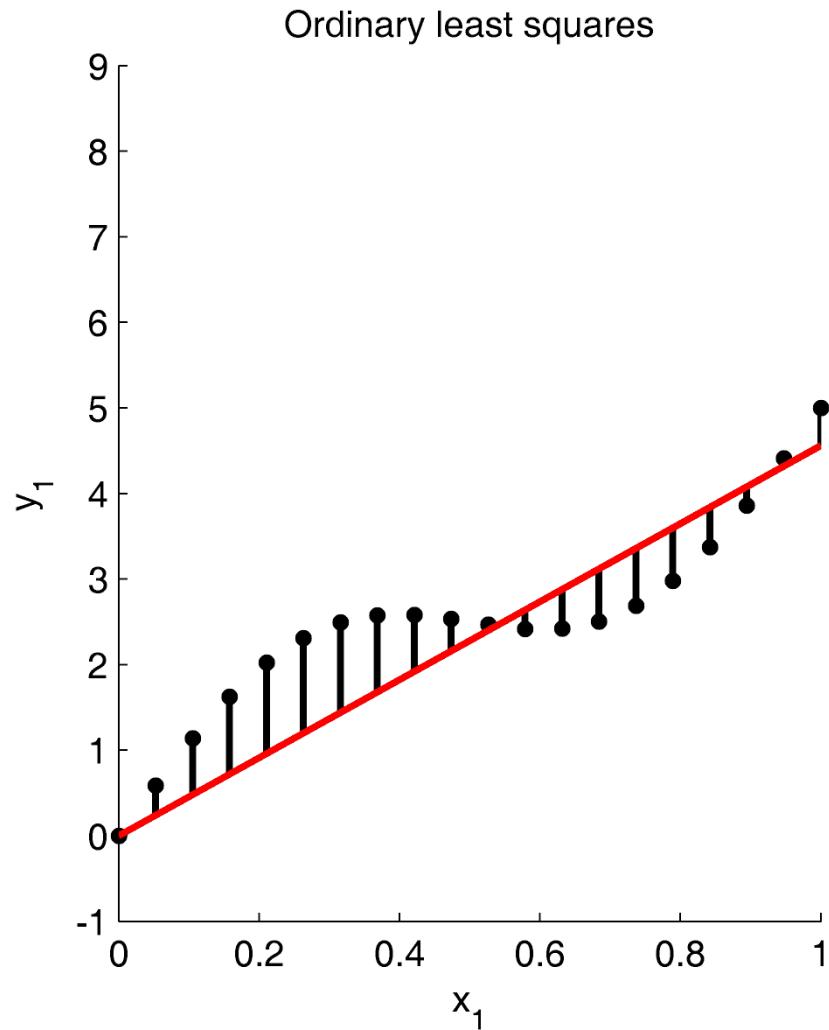
By differentiating with respect to \mathbf{A} and equating to zero

$$-2\mathbf{X}^\top \mathbf{W} \mathbf{Y} + 2\mathbf{X}^\top \mathbf{W} \mathbf{X} \mathbf{A} = \mathbf{0} \iff \hat{\mathbf{A}} = \boxed{(\mathbf{X}^\top \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{W} \mathbf{Y}}$$

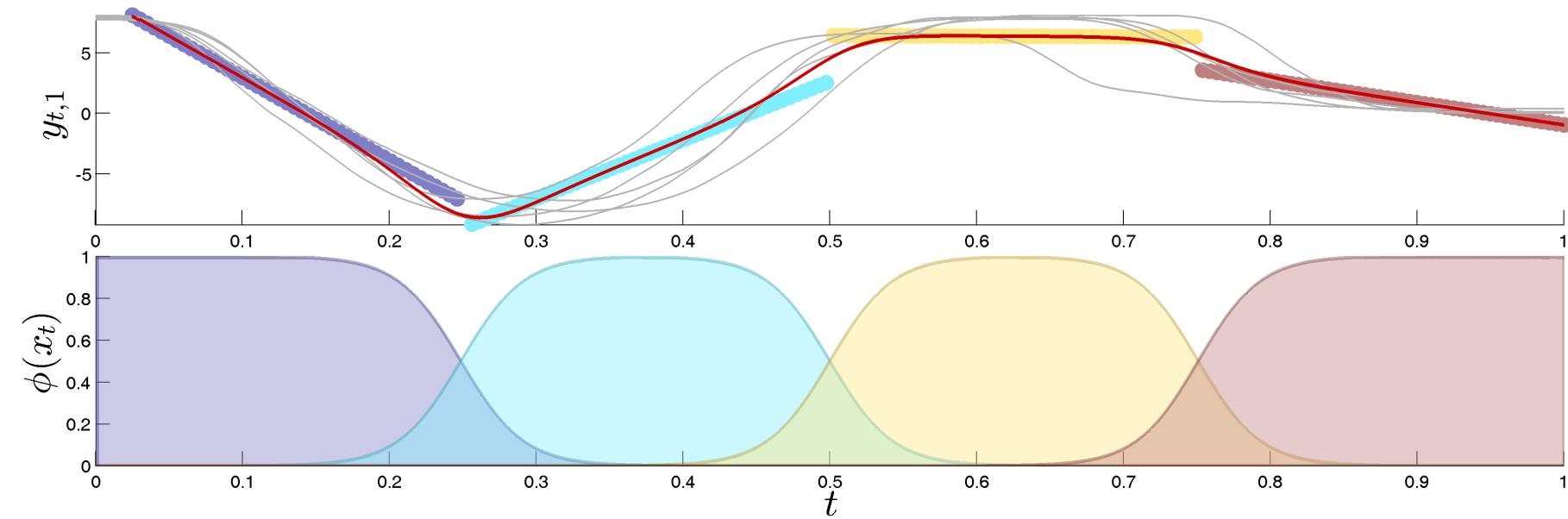
$\mathbf{X}_\mathbf{W}^\dagger$

Weighted least squares

$$\hat{\mathbf{A}} = (\mathbf{X}^\top \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{W} \mathbf{Y}$$



Locally weighted regression (LWR)

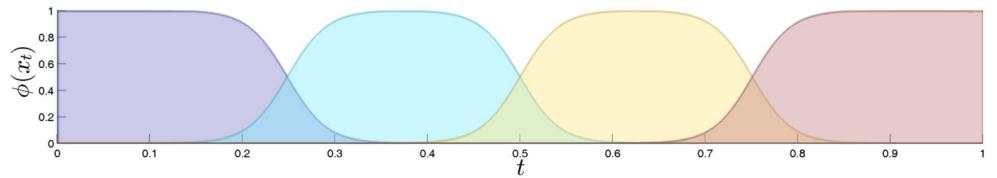


Locally weighted regression (LWR)

LWR computes K estimates $\hat{\mathbf{A}}_k$, each with a different weighting function $\phi_k(\mathbf{x}_n^{\mathcal{I}})$, often defined as the **radial basis functions** (RBF)

$$\tilde{\phi}_k(\mathbf{x}_n^{\mathcal{I}}) = \exp\left(-\frac{1}{2}(\mathbf{x}_n^{\mathcal{I}} - \boldsymbol{\mu}_k^{\mathcal{I}})^{\top} \boldsymbol{\Sigma}_k^{\mathcal{I}}^{-1} (\mathbf{x}_n^{\mathcal{I}} - \boldsymbol{\mu}_k^{\mathcal{I}})\right),$$

or in its rescaled form as



$$\phi_k(\mathbf{x}_n^{\mathcal{I}}) = \frac{\tilde{\phi}_k(\mathbf{x}_n^{\mathcal{I}})}{\sum_{i=1}^K \tilde{\phi}_i(\mathbf{x}_n^{\mathcal{I}})},$$

where $\boldsymbol{\mu}_k^{\mathcal{I}}$ and $\boldsymbol{\Sigma}_k^{\mathcal{I}}$ are the parameters of the k -th RBF.

- K weighted regressions on the same dataset $\{\mathbf{X}^{\mathcal{I}}, \mathbf{X}^{\mathcal{O}}\}$
- Nonlinear problem solved locally by linear regression

Locally weighted regression (LWR)

Often, the centroids $\mu_k^{\mathcal{I}}$ are set to uniformly cover the input space, and $\Sigma_k^{\mathcal{I}} = \mathbf{I}\sigma^2$ is used as a common bandwidth shared by all basis functions.

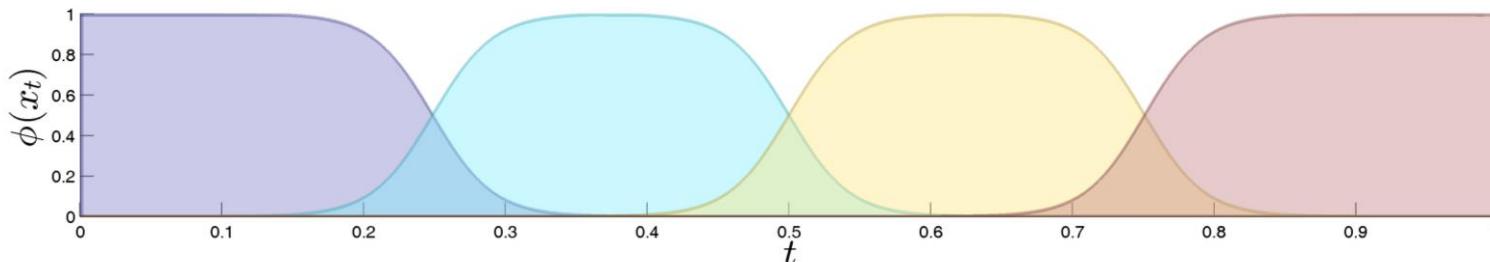
An associated diagonal matrix

$$\mathbf{X}^{\mathcal{I}} = [t_1, t_2, \dots, t_N]^{\top}$$
$$\hat{\mathbf{A}}_k = (\mathbf{X}^{\mathcal{I}\top} \mathbf{W}_k \mathbf{X}^{\mathcal{I}})^{-1} \mathbf{X}^{\mathcal{I}\top} \mathbf{W}_k \mathbf{X}^{\mathcal{O}}$$

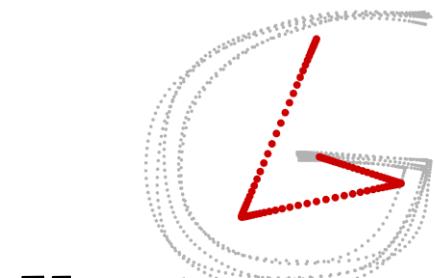
$$\mathbf{W}_k = \text{diag}\left(\phi_k(\mathbf{x}_1^{\mathcal{I}}), \phi_k(\mathbf{x}_2^{\mathcal{I}}), \dots, \phi_k(\mathbf{x}_N^{\mathcal{I}})\right)$$

can be used to evaluate $\hat{\mathbf{A}}_k$. The result can then be used to compute

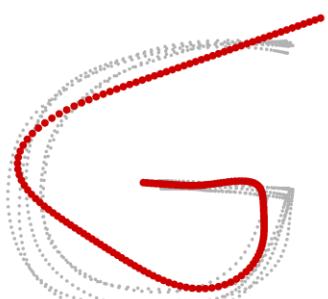
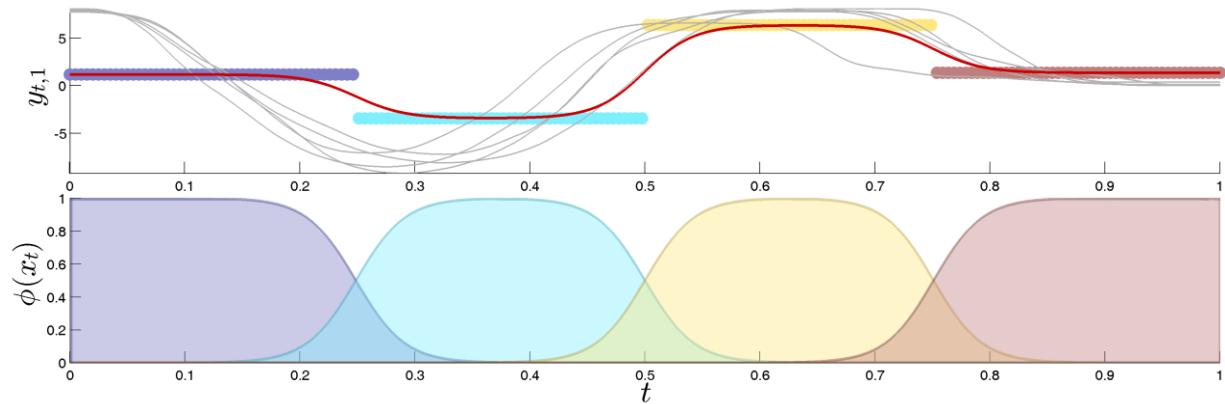
$$\mathbf{X}^{\mathcal{O}} = \sum_{k=1}^K \mathbf{W}_k \mathbf{X}^{\mathcal{I}} \hat{\mathbf{A}}_k$$



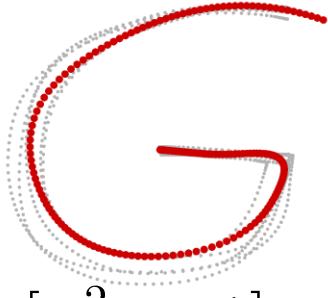
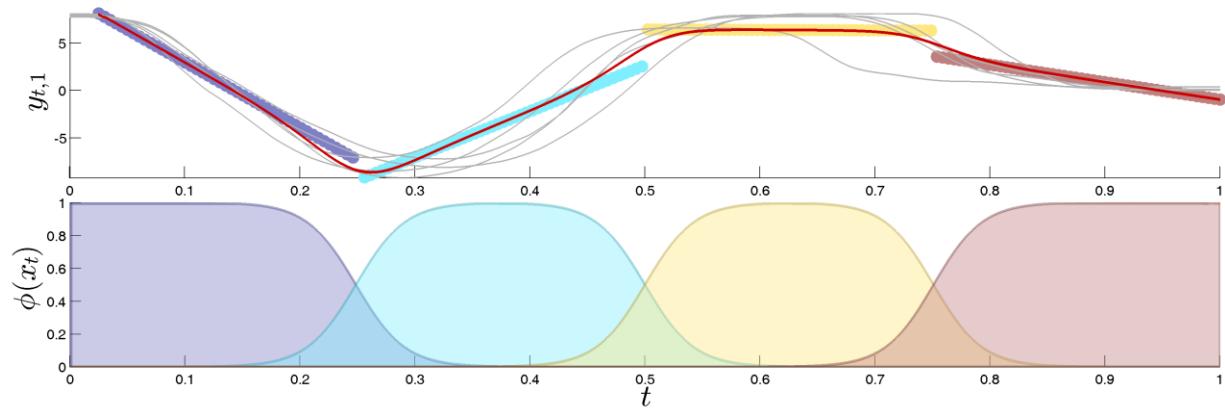
Locally weighted regression (LWR)



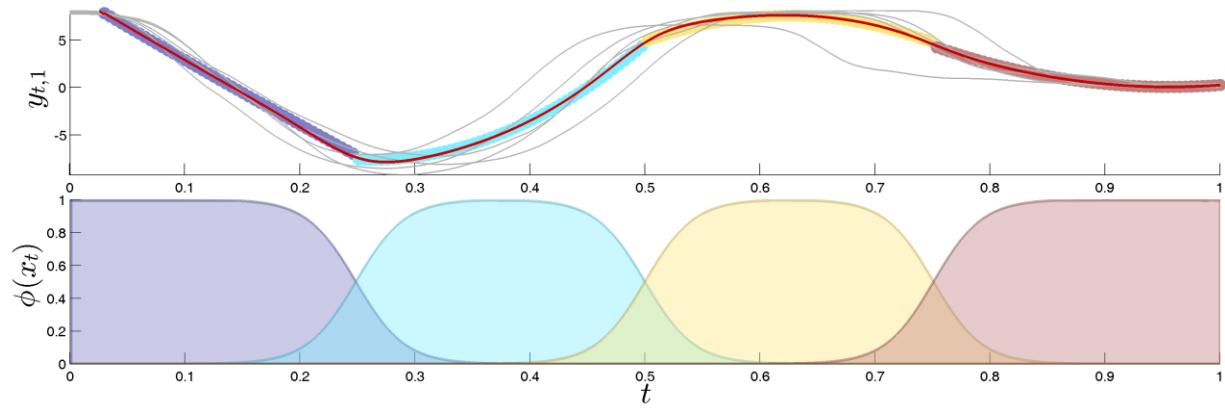
$$X = 1$$



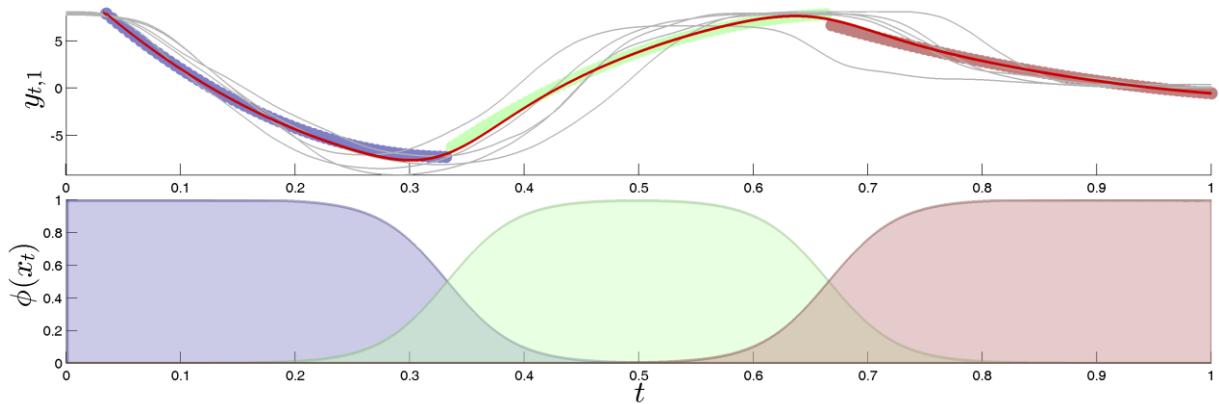
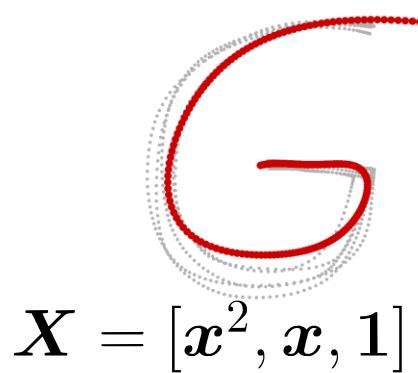
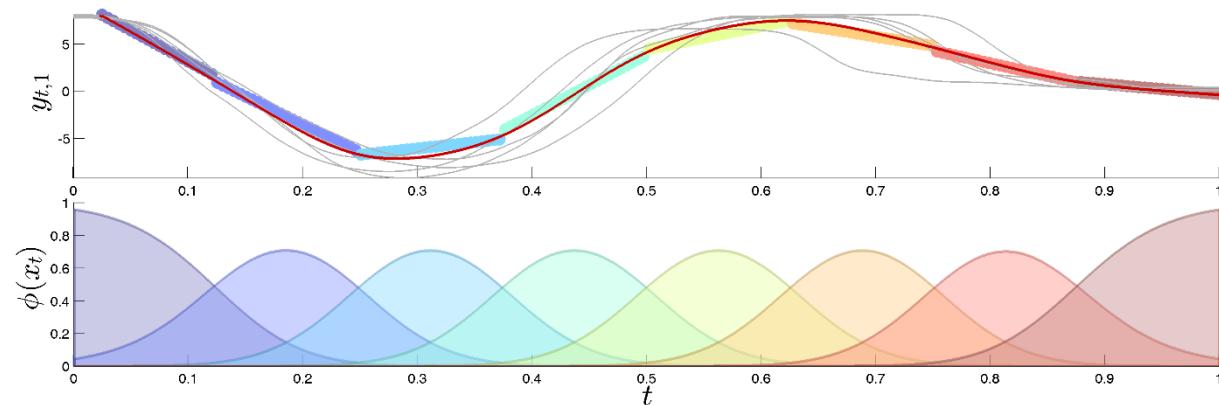
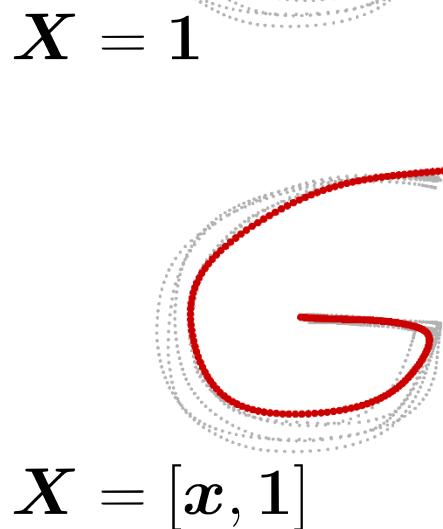
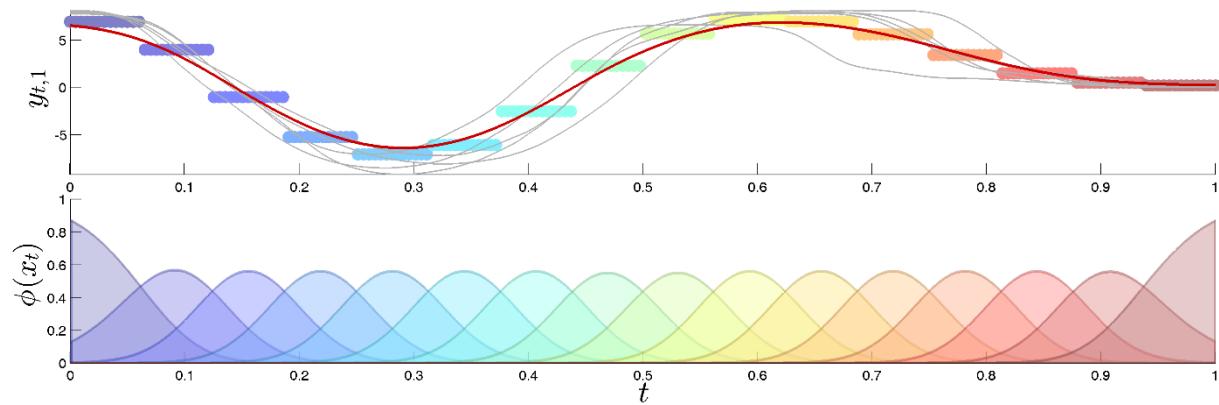
$$X = [x, 1]$$



$$X = [x^2, x, 1]$$



Locally weighted regression (LWR)



Locally weighted regression (LWR) - Resources

Softwares

<http://www.idiap.ch/software/pbdlib/>

Matlab codes: demo_LWR01.m

C++ codes: demo_LWR_batch01.cpp

References

[Atkeson, Moore and Schaal, “*Locally weighted learning for control*”, Artificial Intelligence Review, 11(1-5), 1997]

[Cleveland, “*Robust locally weighted regression and smoothing scatterplots*”, American Statistical Association 74(368), 1979]

[Calinon and Lee, “*Learning Control*”, Humanoid Robotics: a Reference (Springer), 2019]

Outline

- **Superposition with basis functions**

- Bezier curves

- Locally weighted regression (LWR)

- Gaussian mixture regression (GMR)**

- Fourier series for periodic motion and ergodic control

- **Dynamical movement primitives (DMP)**

- Probabilistic movement primitives (ProMP)

- **Superposition Vs fusion**

- Product of Gaussians

- **Model predictive control (MPC)**

- Linear quadratic tracking (LQT)

- Task-parameterized movement models

- **Differential geometry**

- Riemannian manifolds

Gaussian Mixture Model (GMM)

K Gaussians
 N datapoints of dimension D

$$\mathcal{P}(\boldsymbol{\xi}_t) = \sum_{i=1}^K \pi_i \mathcal{N}(\boldsymbol{\xi}_t | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$$

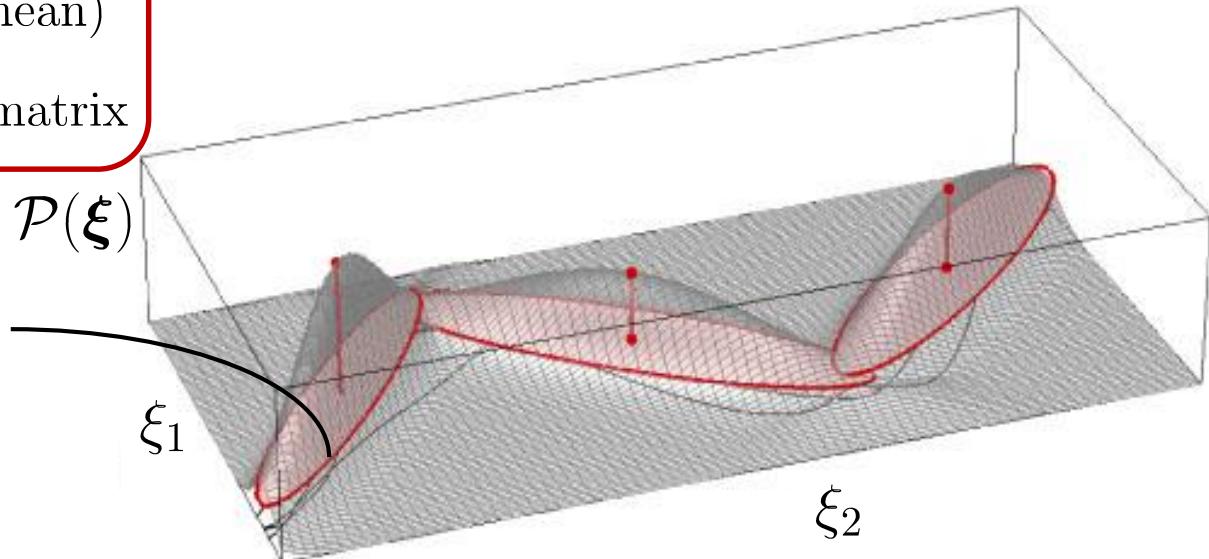
$$\mathcal{N}(\boldsymbol{\xi}_t | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \frac{1}{(2\pi)^{\frac{D}{2}} |\boldsymbol{\Sigma}_i|^{\frac{1}{2}}} \exp \left(-\frac{1}{2} (\boldsymbol{\xi}_t - \boldsymbol{\mu}_i)^\top \boldsymbol{\Sigma}_i^{-1} (\boldsymbol{\xi}_t - \boldsymbol{\mu}_i) \right)$$

$\boldsymbol{\xi} \in \mathbb{R}^{D \times N}$ Observations ($N = \sum_{m=1}^M T_m$, the m -th trajectory has T_m datapoints)

$\pi_i \in \mathbb{R}$	Mixing coefficient
$\boldsymbol{\mu}_i \in \mathbb{R}^D$	Center (or mean)
$\boldsymbol{\Sigma}_i \in \mathbb{R}^{D \times D}$	Covariance matrix

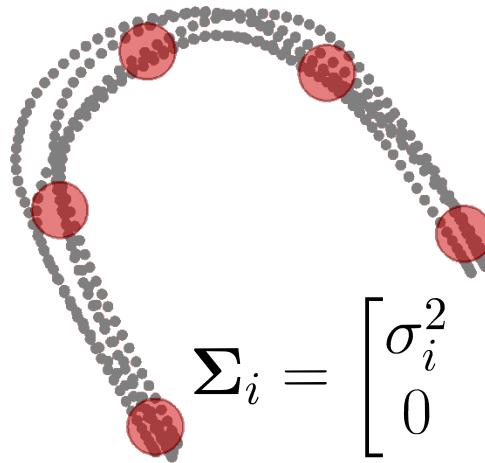
Parameters $\boldsymbol{\Theta}^{\text{GMM}} = \{\pi_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^K$

Equidensity contour of one standard deviation



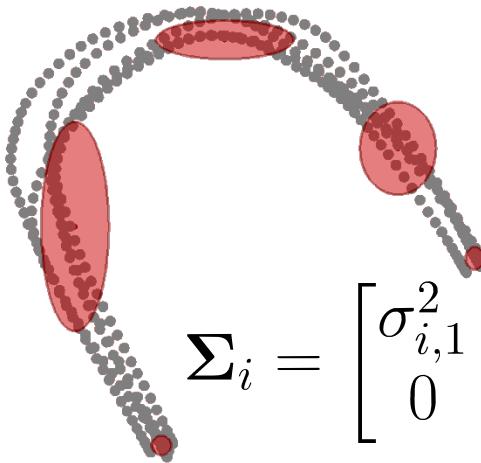
Covariance structures in GMM

Isotropic



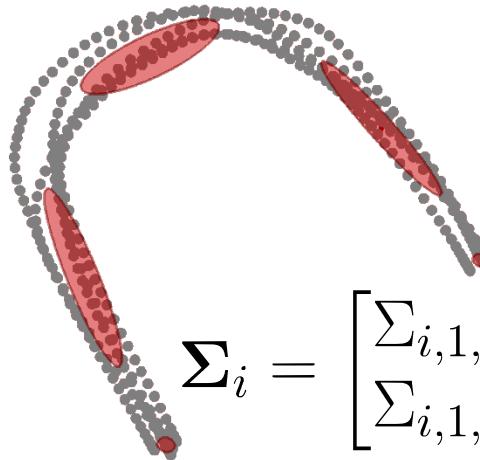
$$\Sigma_i = \begin{bmatrix} \sigma_i^2 & 0 \\ 0 & \sigma_i^2 \end{bmatrix}$$

Diagonal



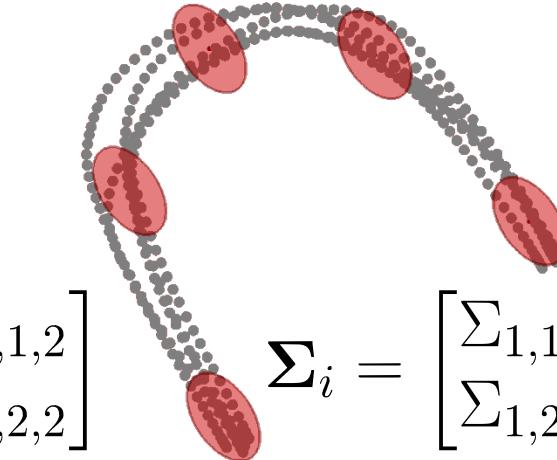
$$\Sigma_i = \begin{bmatrix} \sigma_{i,1}^2 & 0 \\ 0 & \sigma_{i,2}^2 \end{bmatrix}$$

Full



$$\Sigma_i = \begin{bmatrix} \Sigma_{i,1,1} & \Sigma_{i,1,2} \\ \Sigma_{i,1,2} & \Sigma_{i,2,2} \end{bmatrix}$$

Tied



$$\Sigma_i = \begin{bmatrix} \Sigma_{1,1} & \Sigma_{1,2} \\ \Sigma_{1,2} & \Sigma_{2,2} \end{bmatrix}$$

Parameters estimation in GMM... in 1893

54 pages!

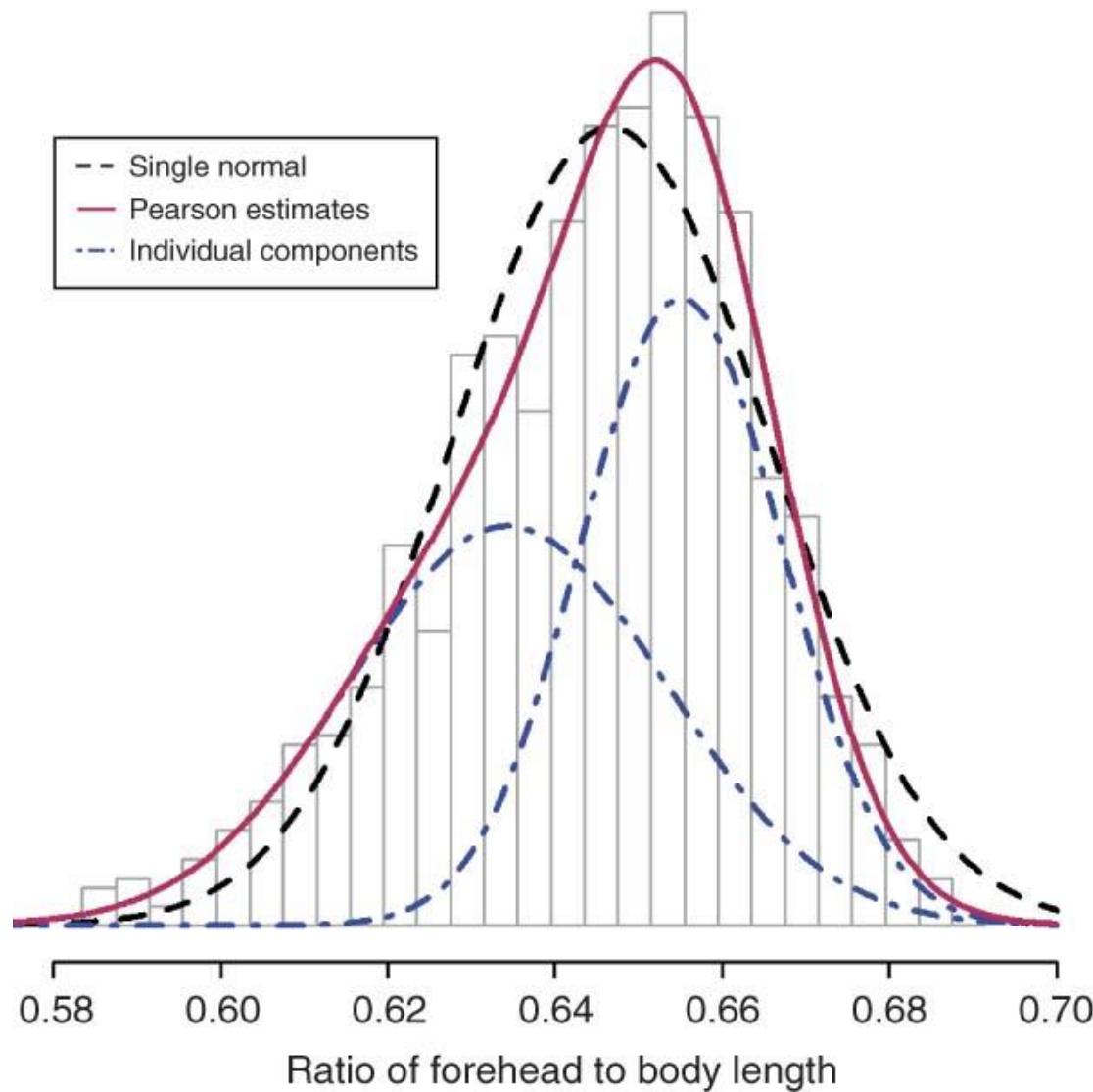
Proposed solution:
Moment-based approach
requiring to solve a
polynomial of degree 9...

... which does not mean that moment-based approaches are old-fashioned!

They are actually today popular again with new developments related to spectral decomposition.

Parameters estimation in GMM... in 1893

Pearson's crab data and fitted mixtures



EM for GMM: Resulting procedure

K Gaussians
N datapoints

E-step:

$$h_{t,i} = \frac{\pi_i \mathcal{N}(\xi_t | \mu_i, \Sigma_i)}{\sum_{k=1}^K \pi_k \mathcal{N}(\xi_t | \mu_k, \Sigma_k)}$$

M-step:

$$\pi_i \leftarrow \frac{\sum_{t=1}^N h_{t,i}}{N},$$

$$\mu_i \leftarrow \frac{\sum_{t=1}^N h_{t,i} \xi_t}{\sum_{t=1}^N h_{t,i}},$$

$$\Sigma_i \leftarrow \frac{\sum_{t=1}^N h_{t,i} (\xi_t - \mu_i)(\xi_t - \mu_i)^\top}{\sum_{t=1}^N h_{t,i}}$$

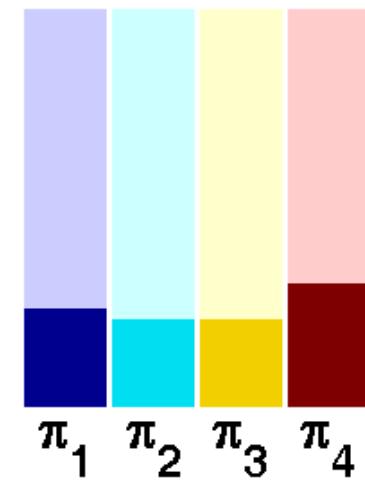
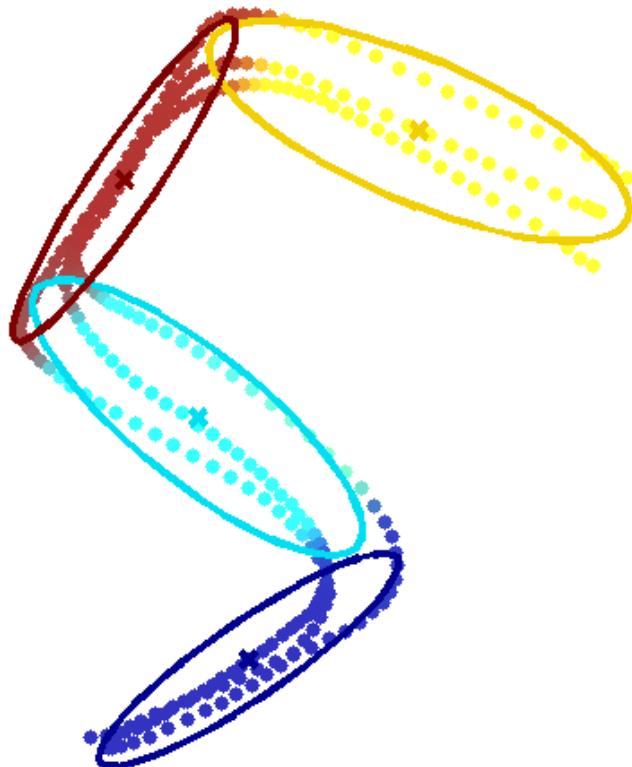
These results can be intuitively interpreted in terms of normalized counts.

EM provides a systematic approach to derive such procedure.

→ **Weighted averages taking into account the responsibility of each datapoint in each cluster.**

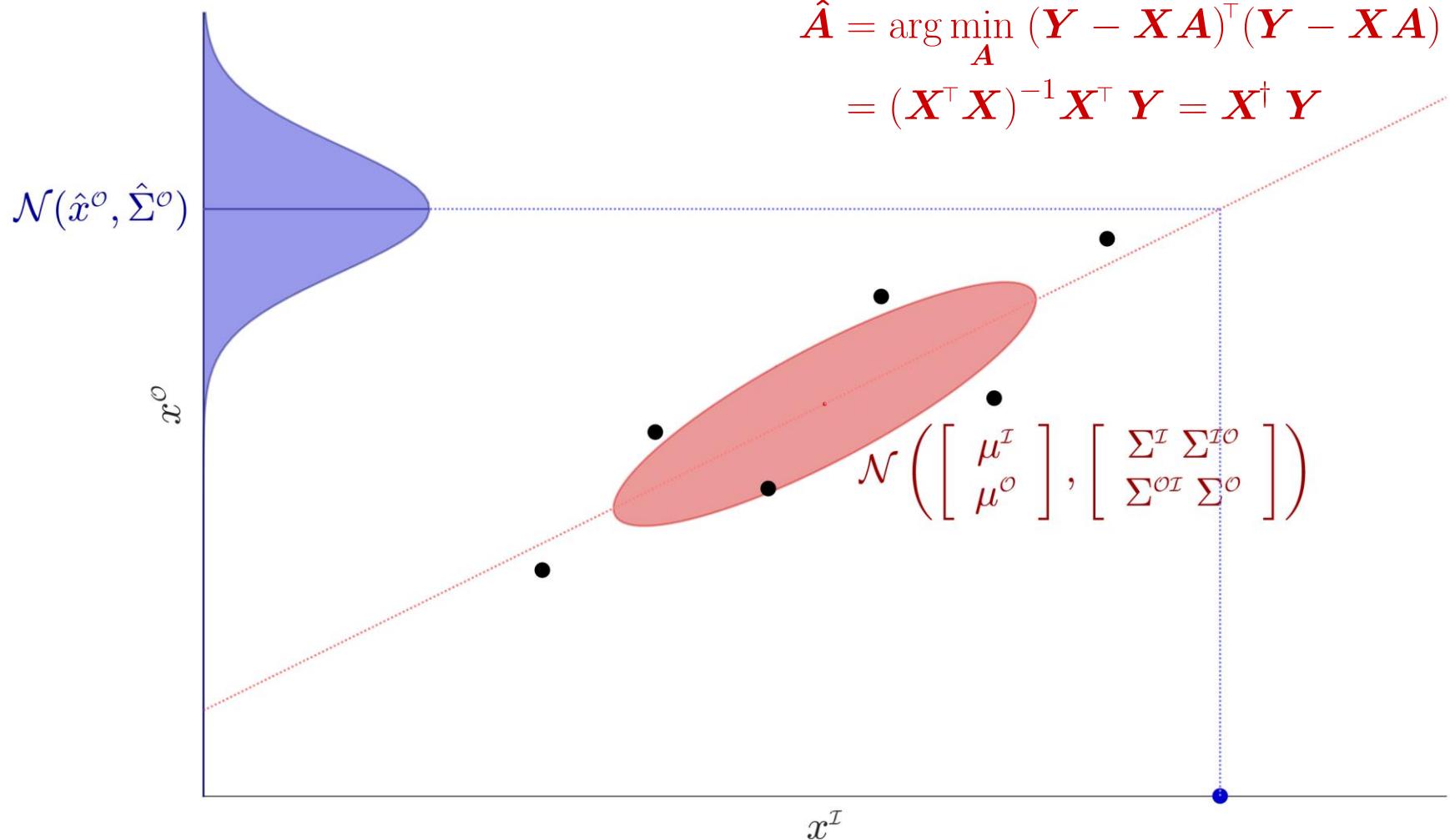
EM for GMM

M-step



Gaussian conditioning (regression from joint distribution)

$$\begin{aligned}\hat{\mathbf{A}} &= \arg \min_{\mathbf{A}} (\mathbf{Y} - \mathbf{X} \mathbf{A})^\top (\mathbf{Y} - \mathbf{X} \mathbf{A}) \\ &= (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y} = \mathbf{X}^\dagger \mathbf{Y}\end{aligned}$$



→ Linear regression from joint distribution

Gaussian conditioning

We consider multivariate datapoints \mathbf{x} and multivariate Gaussian distributions characterized by centers $\boldsymbol{\mu}$ and covariances $\boldsymbol{\Sigma}$, that can be partitioned as

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}^{\mathcal{I}} \\ \mathbf{x}^{\mathcal{O}} \end{bmatrix}, \quad \boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}^{\mathcal{I}} \\ \boldsymbol{\mu}^{\mathcal{O}} \end{bmatrix}, \quad \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}^{\mathcal{I}} & \boldsymbol{\Sigma}^{\mathcal{I}\mathcal{O}} \\ \boldsymbol{\Sigma}^{\mathcal{O}\mathcal{I}} & \boldsymbol{\Sigma}^{\mathcal{O}} \end{bmatrix}.$$

If $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, we have that $\mathbf{x}^{\mathcal{O}} | \mathbf{x}^{\mathcal{I}} \sim \mathcal{N}(\hat{\mathbf{x}}^{\mathcal{O}}, \hat{\boldsymbol{\Sigma}}^{\mathcal{O}})$, with parameters

$$\begin{aligned} \hat{\mathbf{x}}^{\mathcal{O}} &= \boldsymbol{\mu}^{\mathcal{O}} + \boldsymbol{\Sigma}^{\mathcal{O}\mathcal{I}} \boldsymbol{\Sigma}^{\mathcal{I}}^{-1} (\mathbf{x}^{\mathcal{I}} - \boldsymbol{\mu}^{\mathcal{I}}), \\ \hat{\boldsymbol{\Sigma}}^{\mathcal{O}} &= \boldsymbol{\Sigma}^{\mathcal{O}} - \boldsymbol{\Sigma}^{\mathcal{O}\mathcal{I}} \boldsymbol{\Sigma}^{\mathcal{I}}^{-1} \boldsymbol{\Sigma}^{\mathcal{I}\mathcal{O}}. \end{aligned}$$

We can see that $\hat{\mathbf{x}}^{\mathcal{O}}$ is linearly dependent on $\mathbf{x}^{\mathcal{I}}$, and that $\hat{\boldsymbol{\Sigma}}^{\mathcal{O}}$ is independent of $\mathbf{x}^{\mathcal{I}}$.

We can also notice that for full joint covariance, the conditional covariance $\hat{\boldsymbol{\Sigma}}^{\mathcal{O}}$ will typically be smaller than the marginal $\boldsymbol{\Sigma}^{\mathcal{O}}$.

Gaussian mixture regression (GMR)

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}^{\mathcal{I}} \\ \boldsymbol{x}^{\mathcal{O}} \end{bmatrix} \quad \boldsymbol{\mu}_i = \begin{bmatrix} \boldsymbol{\mu}_i^{\mathcal{I}} \\ \boldsymbol{\mu}_i^{\mathcal{O}} \end{bmatrix} \quad \boldsymbol{\Sigma}_i = \begin{bmatrix} \boldsymbol{\Sigma}_i^{\mathcal{I}} & \boldsymbol{\Sigma}_i^{\mathcal{I}\mathcal{O}} \\ \boldsymbol{\Sigma}_i^{\mathcal{O}\mathcal{I}} & \boldsymbol{\Sigma}_i^{\mathcal{O}} \end{bmatrix}$$

$\mathcal{P}(\boldsymbol{x}^{\mathcal{O}}|\boldsymbol{x}^{\mathcal{I}})$ can be computed as the multimodal conditional distribution

$$\mathcal{P}(\boldsymbol{x}^{\mathcal{O}}|\boldsymbol{x}^{\mathcal{I}}) = \sum_{i=1}^K h_i \mathcal{N}\left(\boldsymbol{x}^{\mathcal{O}}|\hat{\boldsymbol{\mu}}_i^{\mathcal{O}}, \hat{\boldsymbol{\Sigma}}_i^{\mathcal{O}}\right),$$

$$\text{with } \hat{\boldsymbol{\mu}}_i^{\mathcal{O}} = \boldsymbol{\mu}_i^{\mathcal{O}} + \boldsymbol{\Sigma}_i^{\mathcal{O}\mathcal{I}} \boldsymbol{\Sigma}_i^{\mathcal{I}}^{-1} (\boldsymbol{x}^{\mathcal{I}} - \boldsymbol{\mu}_i^{\mathcal{I}}),$$

$$\hat{\boldsymbol{\Sigma}}_i^{\mathcal{O}} = \boldsymbol{\Sigma}_i^{\mathcal{O}} - \boldsymbol{\Sigma}_i^{\mathcal{O}\mathcal{I}} \boldsymbol{\Sigma}_i^{\mathcal{I}}^{-1} \boldsymbol{\Sigma}_i^{\mathcal{I}\mathcal{O}}$$

$$\text{and } h_i = \frac{\pi_i \mathcal{N}(\boldsymbol{x}^{\mathcal{I}}|\boldsymbol{\mu}_i^{\mathcal{I}}, \boldsymbol{\Sigma}_i^{\mathcal{I}})}{\sum_k^K \pi_k \mathcal{N}(\boldsymbol{x}^{\mathcal{I}}|\boldsymbol{\mu}_k^{\mathcal{I}}, \boldsymbol{\Sigma}_k^{\mathcal{I}})},$$

computed with the marginal

$$\mathcal{N}(\boldsymbol{x}^{\mathcal{I}}|\boldsymbol{\mu}_i^{\mathcal{I}}, \boldsymbol{\Sigma}_i^{\mathcal{I}}) = (2\pi)^{-\frac{D}{2}} |\boldsymbol{\Sigma}_i^{\mathcal{I}}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2} (\boldsymbol{x}^{\mathcal{I}} - \boldsymbol{\mu}_i^{\mathcal{I}})^{\top} \boldsymbol{\Sigma}_i^{\mathcal{I}}^{-1} (\boldsymbol{x}^{\mathcal{I}} - \boldsymbol{\mu}_i^{\mathcal{I}})\right).$$

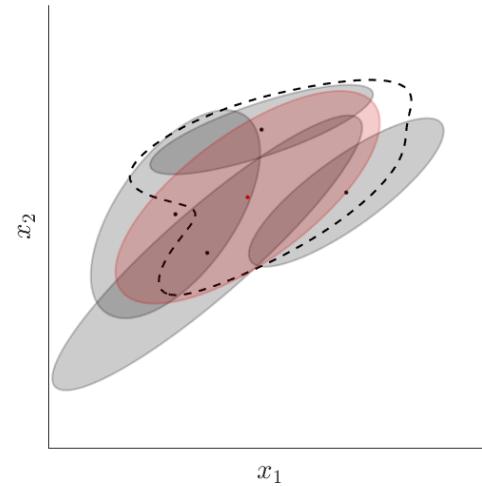
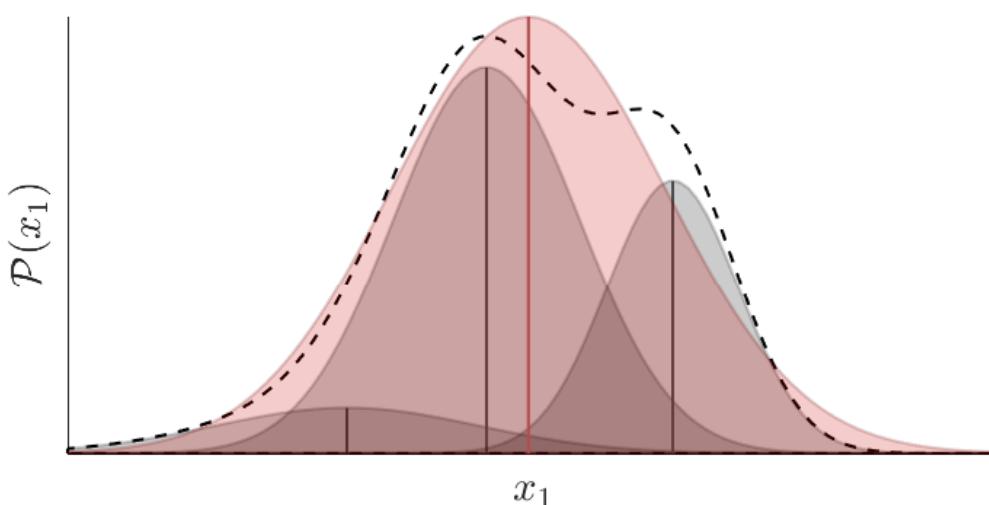
Gaussian estimate of a mixture of Gaussians

We can approximate a mixture of Gaussians $\sum_{i=1}^K h_i \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ with a single Gaussian $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, by **moment matching of the means (first moments) and covariances (second moments)** with

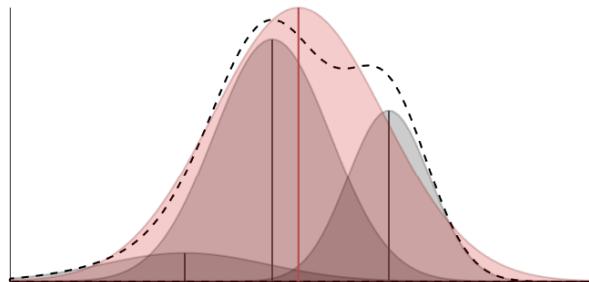
$$\boldsymbol{\mu} = \sum_{i=1}^K h_i \boldsymbol{\mu}_i,$$

$$\boldsymbol{\Sigma} = \sum_{i=1}^K h_i \left(\boldsymbol{\Sigma}_i + \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top \right) - \boldsymbol{\mu} \boldsymbol{\mu}^\top,$$

also referred to as the **law of total mean and (co)variance**.



Gaussian mixture regression (GMR)



$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}^{\mathcal{I}} \\ \boldsymbol{x}^{\mathcal{O}} \end{bmatrix} \quad \boldsymbol{\mu}_i = \begin{bmatrix} \boldsymbol{\mu}_i^{\mathcal{I}} \\ \boldsymbol{\mu}_i^{\mathcal{O}} \end{bmatrix} \quad \boldsymbol{\Sigma}_i = \begin{bmatrix} \boldsymbol{\Sigma}_i^{\mathcal{I}} & \boldsymbol{\Sigma}_i^{\mathcal{I}\mathcal{O}} \\ \boldsymbol{\Sigma}_i^{\mathcal{O}\mathcal{I}} & \boldsymbol{\Sigma}_i^{\mathcal{O}} \end{bmatrix}$$

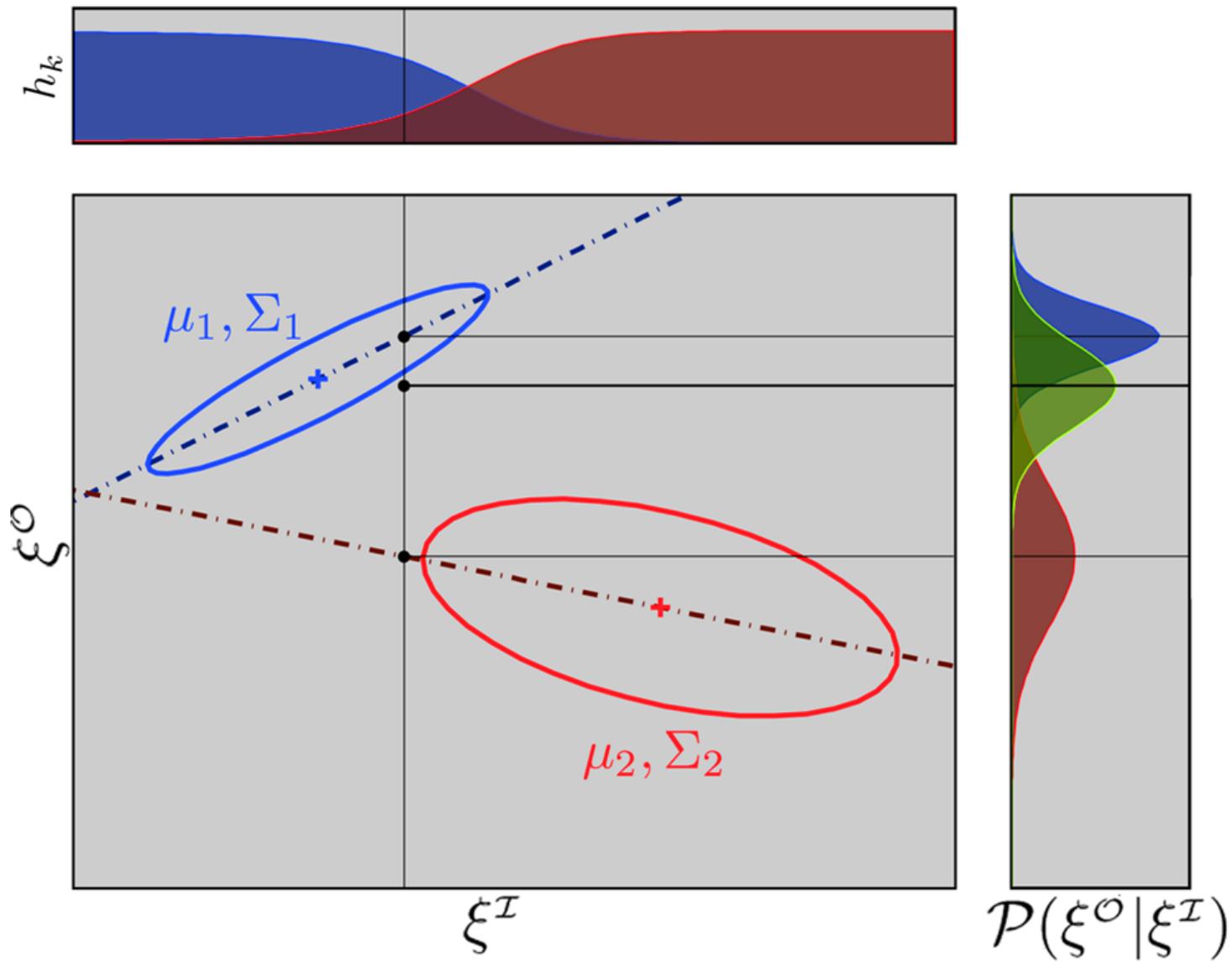
$$\begin{aligned}\hat{\boldsymbol{\mu}}_i^{\mathcal{O}} &= \boldsymbol{\mu}_i^{\mathcal{O}} + \boldsymbol{\Sigma}_i^{\mathcal{O}\mathcal{I}} \boldsymbol{\Sigma}_i^{\mathcal{I}-1} (\boldsymbol{x}^{\mathcal{I}} - \boldsymbol{\mu}_i^{\mathcal{I}}) \\ \hat{\boldsymbol{\Sigma}}_i^{\mathcal{O}} &= \boldsymbol{\Sigma}_i^{\mathcal{O}} - \boldsymbol{\Sigma}_i^{\mathcal{O}\mathcal{I}} \boldsymbol{\Sigma}_i^{\mathcal{I}-1} \boldsymbol{\Sigma}_i^{\mathcal{I}\mathcal{O}}\end{aligned}$$

In GMR, an output distribution as a single multivariate Gaussian can be evaluated by moment matching of the means and covariances. The resulting Gaussian distribution $\mathcal{N}(\hat{\boldsymbol{\mu}}^{\mathcal{O}}, \hat{\boldsymbol{\Sigma}}^{\mathcal{O}})$ has parameters

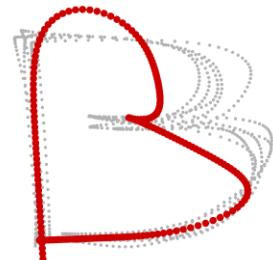
$$\hat{\boldsymbol{\mu}}^{\mathcal{O}} = \sum_{i=1}^K h_i \hat{\boldsymbol{\mu}}_i^{\mathcal{O}},$$

$$\hat{\boldsymbol{\Sigma}}^{\mathcal{O}} = \sum_{i=1}^K h_i \left(\hat{\boldsymbol{\Sigma}}_i^{\mathcal{O}} + \hat{\boldsymbol{\mu}}_i^{\mathcal{O}} \hat{\boldsymbol{\mu}}_i^{\mathcal{O}\top} \right) - \hat{\boldsymbol{\mu}}^{\mathcal{O}} \hat{\boldsymbol{\mu}}^{\mathcal{O}\top}.$$

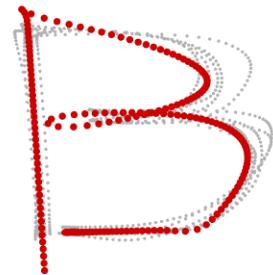
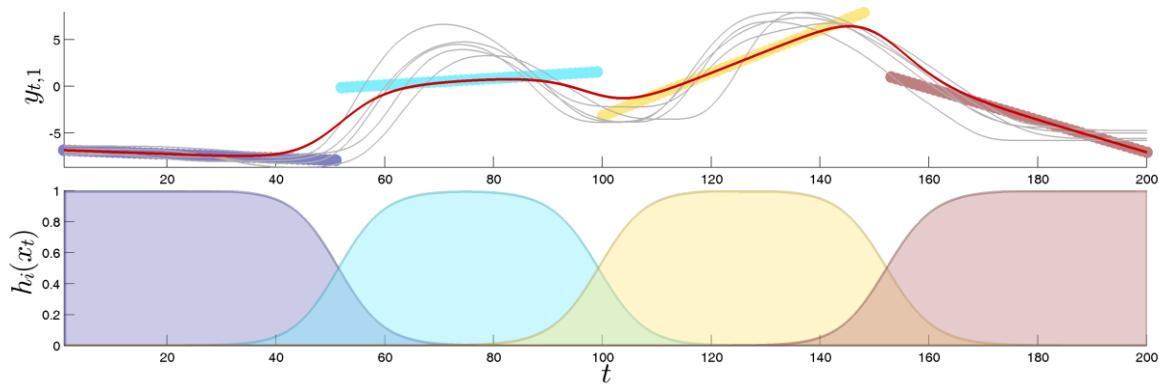
Gaussian mixture regression (GMR)



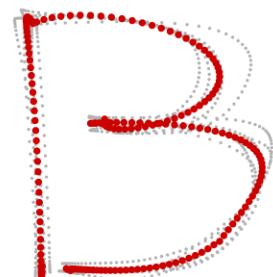
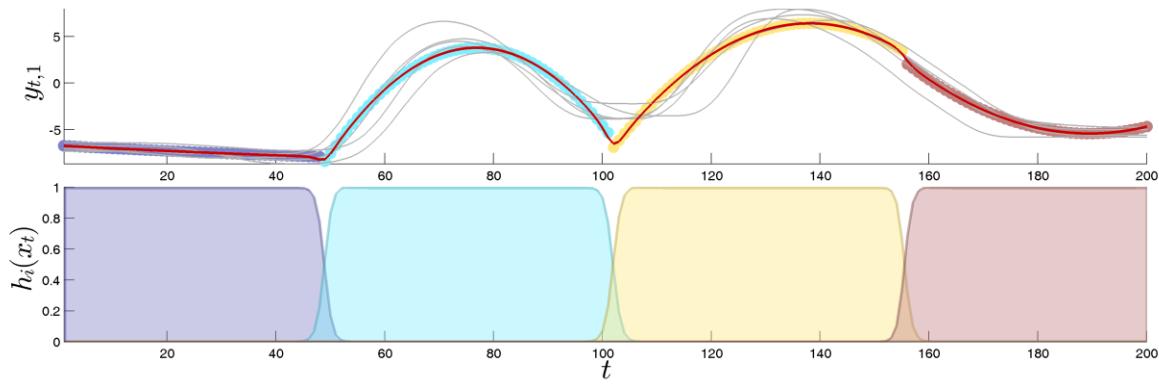
GMR for smooth piecewise polynomial fitting



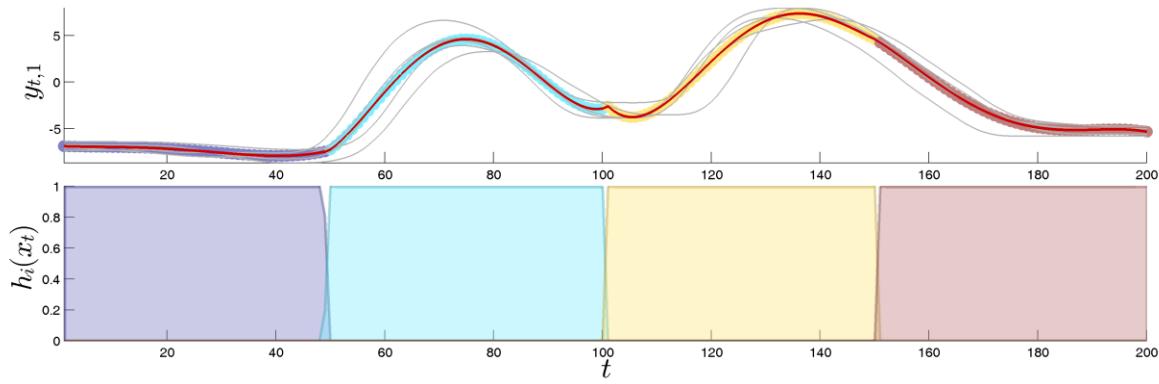
$$X = x$$



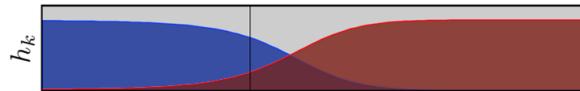
$$X = [x^2, x]$$



$$X = [x^3, x^2, x]$$

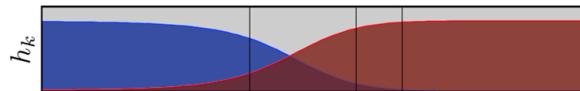
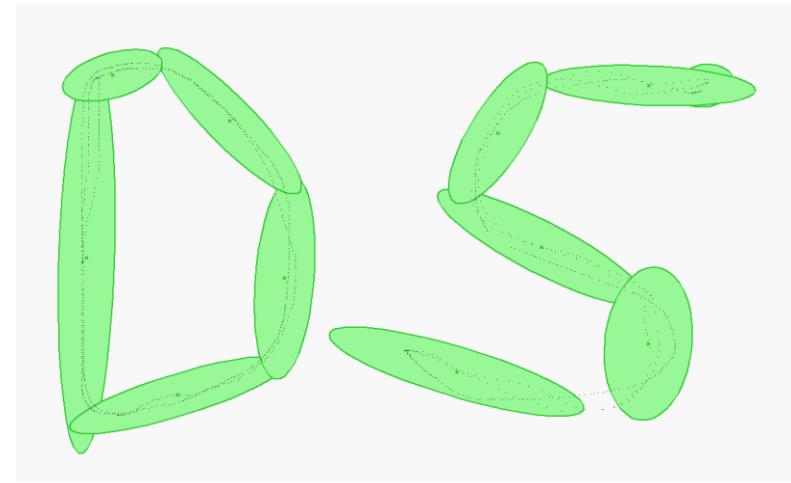
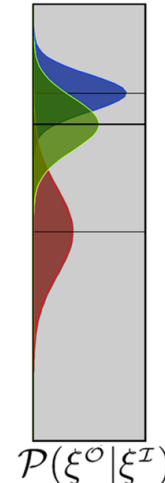
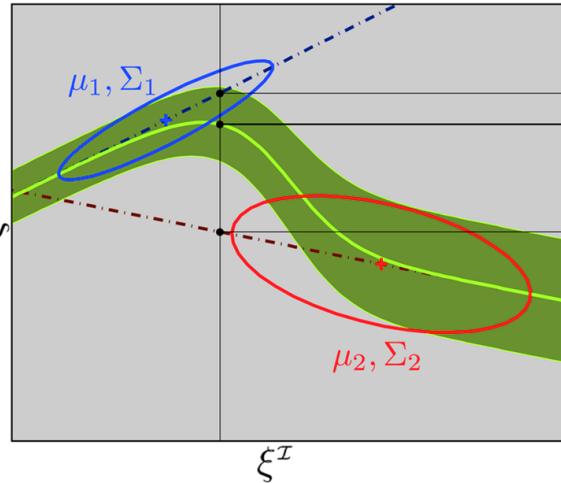


Gaussian mixture regression - Examples

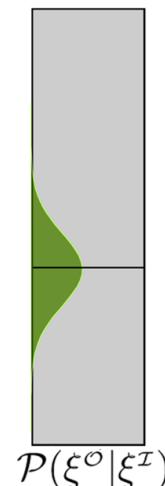
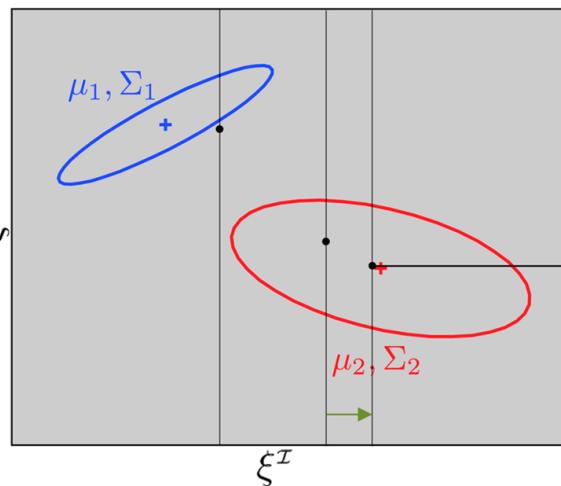


$$\xi^{\mathcal{I}} = \mathbf{t}, \quad \xi^{\mathcal{O}} = \mathbf{x}$$

[Calinon, Guenter and Billard,
IEEE Trans. on SMC-B 37(2), 2007]



$$\xi^{\mathcal{I}} = \mathbf{x}, \quad \xi^{\mathcal{O}} = \dot{\mathbf{x}}$$



With expectation-maximization (EM):
(maximizing log-likelihood)

[Hersch, Guenter, Calinon and Billard,
IEEE Trans. on Robotics 24(6), 2008]

With quadratic programming solver:
(maximizing log-likelihood s.t. stability constraints)

[Khansari-Zadeh and Billard,
IEEE Trans. on Robotics 27(5), 2011]

Gaussian mixture regression (GMR) - Resources

Softwares

<http://www.idiap.ch/software/pbdlib/>

Matlab codes: demo_GMR01.m

C++ codes: demo_GMR01.cpp

References

[Ghahramani and Jordan, “*Supervised learning from incomplete data via an EM approach*”, NIPS’1994]

[Calinon, “*A tutorial on task-parameterized movement learning and retrieval*”, Intelligent Service Robotics 9(1), 2016]

[Calinon and Lee, “*Learning Control*”, Humanoid Robotics: a Reference (Springer), 2019]

Outline

- **Superposition with basis functions**

- Bezier curves

- Locally weighted regression (LWR)

- Gaussian mixture regression (GMR)

- Fourier series for periodic motion and ergodic control**

- **Dynamical movement primitives (DMP)**

- Probabilistic movement primitives (ProMP)

- **Superposition Vs fusion**

- Product of Gaussians

- **Model predictive control (MPC)**

- Linear quadratic tracking (LQT)

- Task-parameterized movement models

- **Differential geometry**

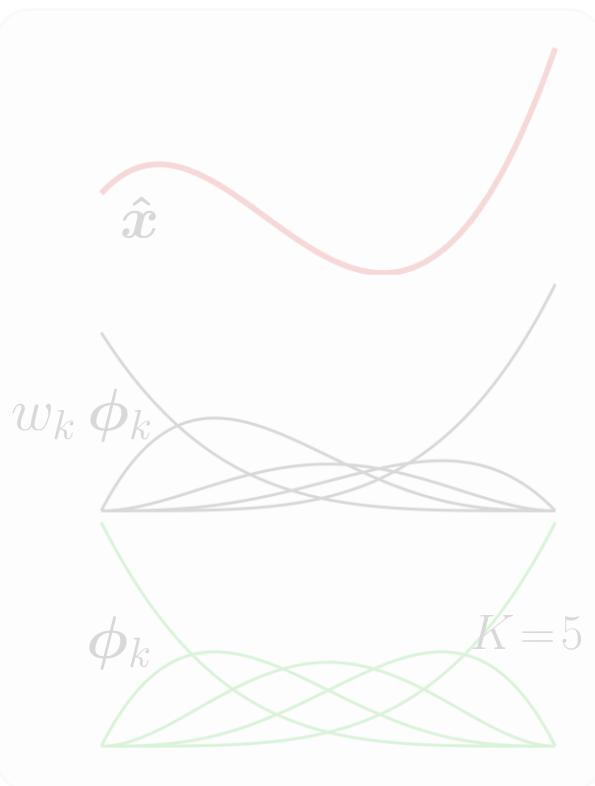
- Riemannian manifolds

Superposition with basis functions

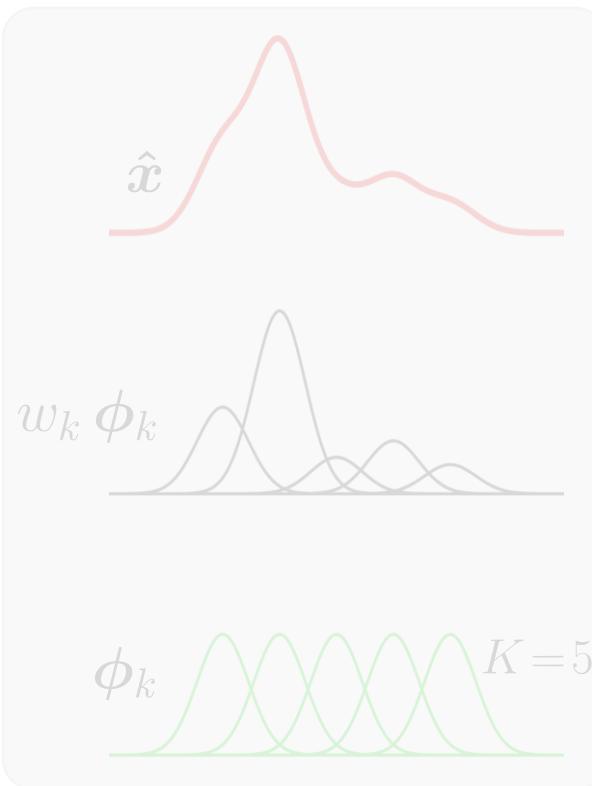
$$\hat{x} = \sum_{k=1}^K w_k \phi_k$$



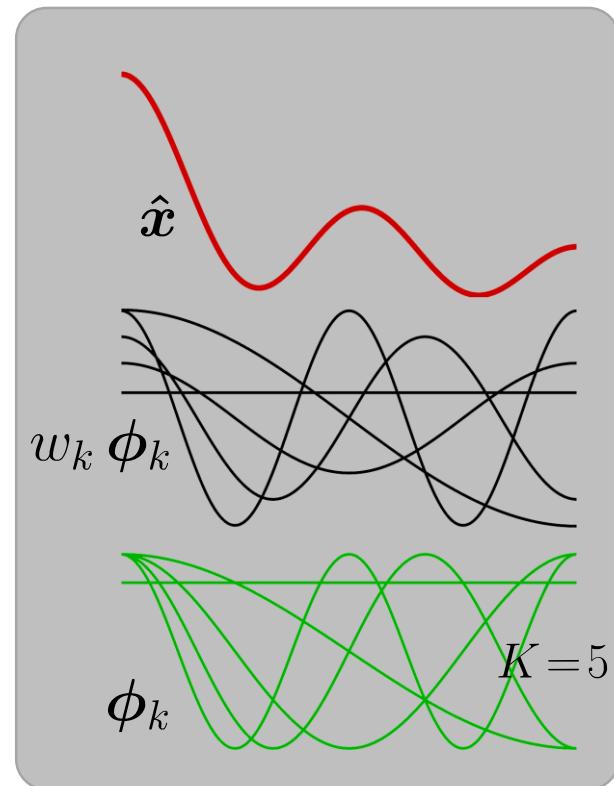
Bézier curves
(Bernstein bases)



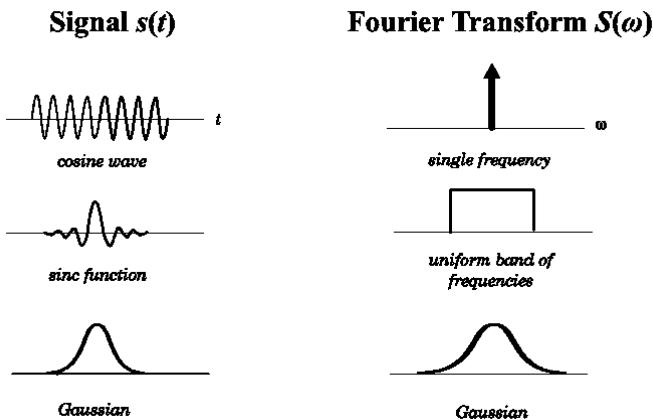
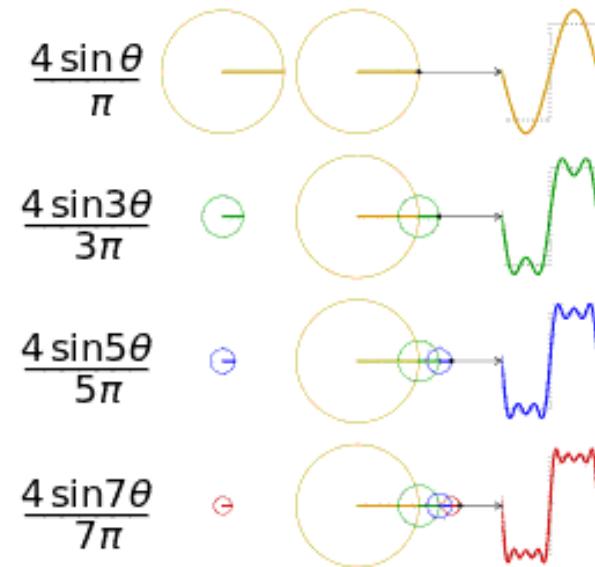
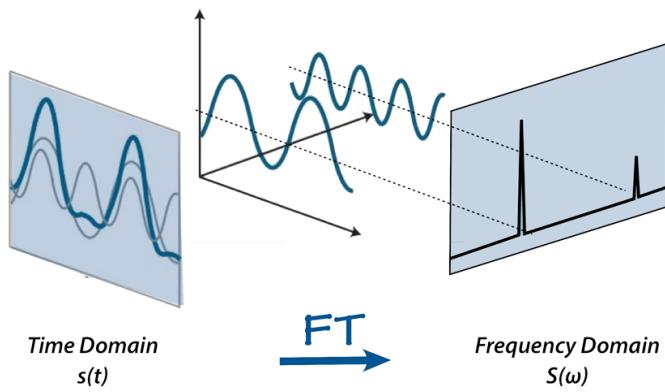
Locally weighted
regression (radial bases)



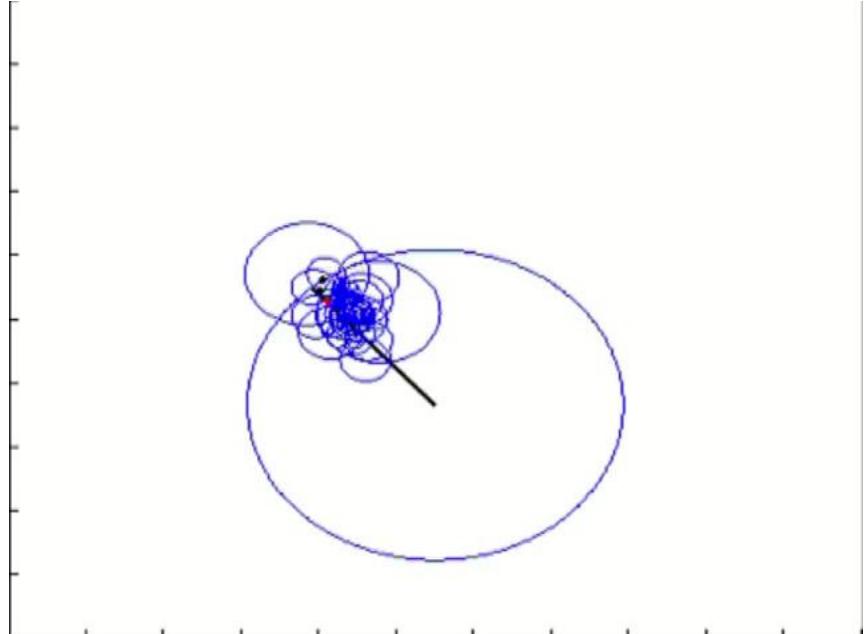
Fourier series
(cosine bases)



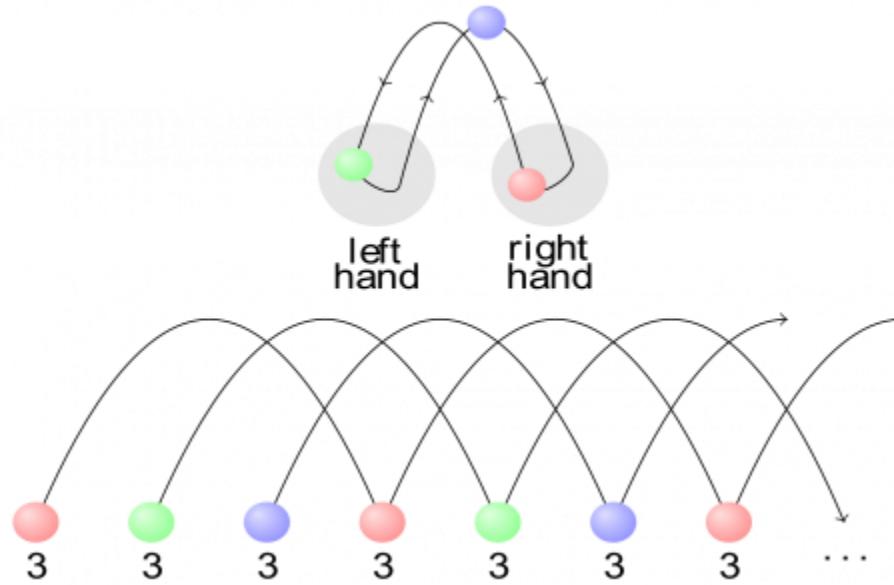
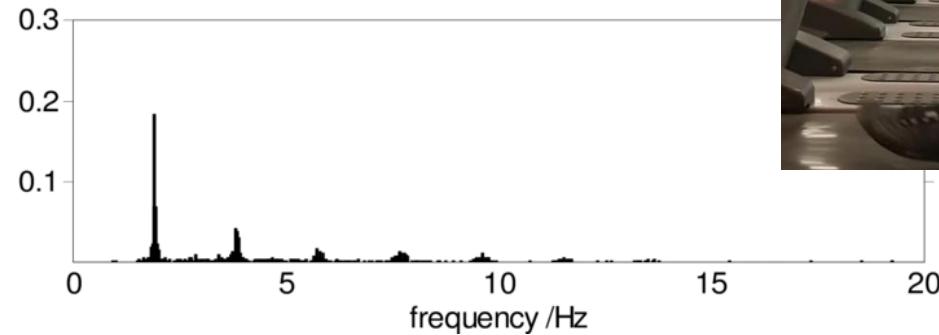
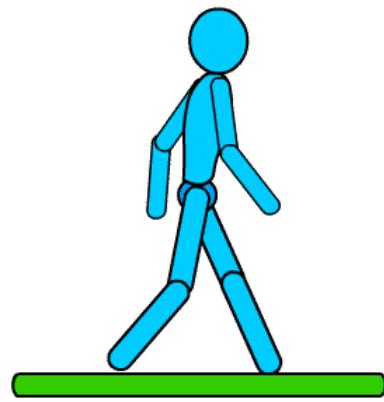
Fourier series



$$\begin{aligned}
 f_k(x) &= \exp\left(i \frac{2\pi k x}{L}\right) \\
 &= \cos\left(\frac{2\pi k x}{L}\right) + i \sin\left(\frac{2\pi k x}{L}\right)
 \end{aligned}$$



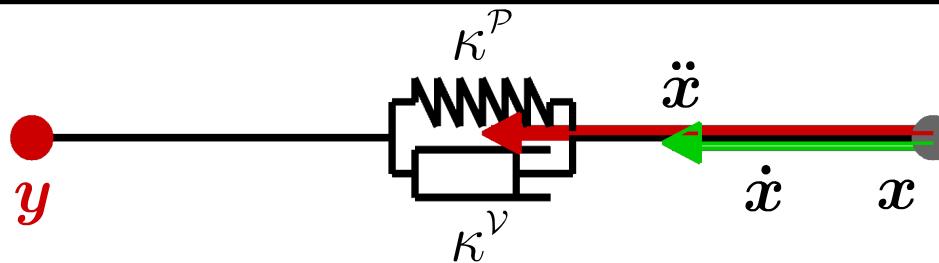
Periodic movements



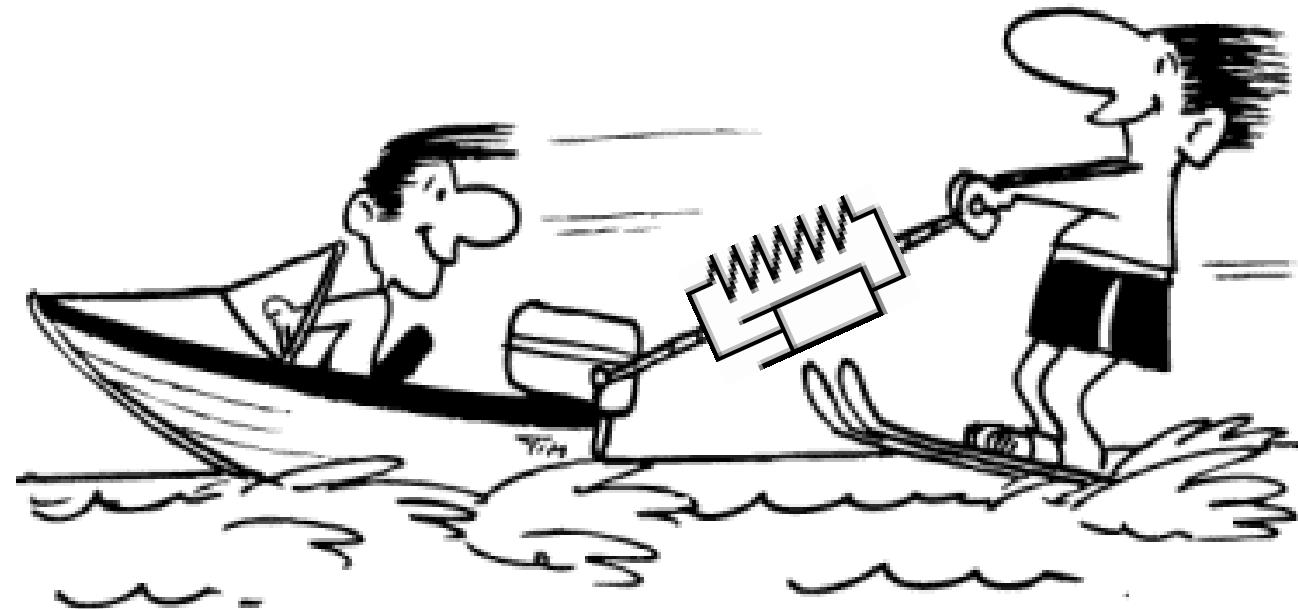
Outline

- Superposition with basis functions
 - Bezier curves
 - Locally weighted regression (LWR)
 - Gaussian mixture regression (GMR)
 - Fourier series for periodic motion and ergodic control
- **Dynamical movement primitives (DMP)**
 - Probabilistic movement primitives (ProMP)
- Superposition Vs fusion
 - Product of Gaussians
- Model predictive control (MPC)
 - Linear quadratic tracking (LQT)
 - Task-parameterized movement models
- Differential geometry
 - Riemannian manifolds

Spring-damper system



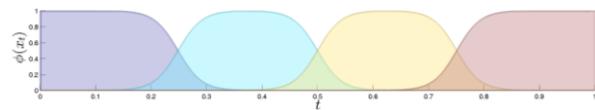
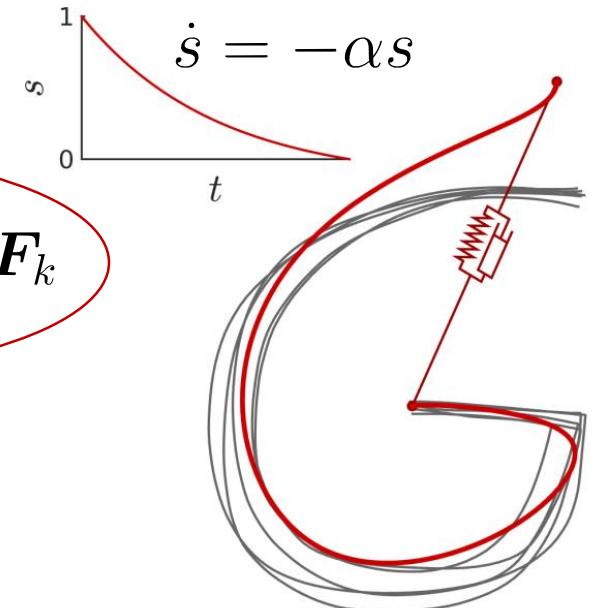
$$\ddot{x} = \kappa^P[y - x] - \kappa^v \dot{x}$$
$$\Rightarrow y = \frac{1}{\kappa^P} \ddot{x} + \frac{\kappa^v}{\kappa^P} \dot{x} + x$$



Dynamical movement primitives (DMP)

$$\ddot{\boldsymbol{x}} = k^p(\boldsymbol{\mu}_T - \boldsymbol{x}) - k^\nu \dot{\boldsymbol{x}} + \boldsymbol{f}(s)$$

$$\boldsymbol{f}(s) = s \sum_{k=1}^K \phi_k(s) \boldsymbol{F}_k$$



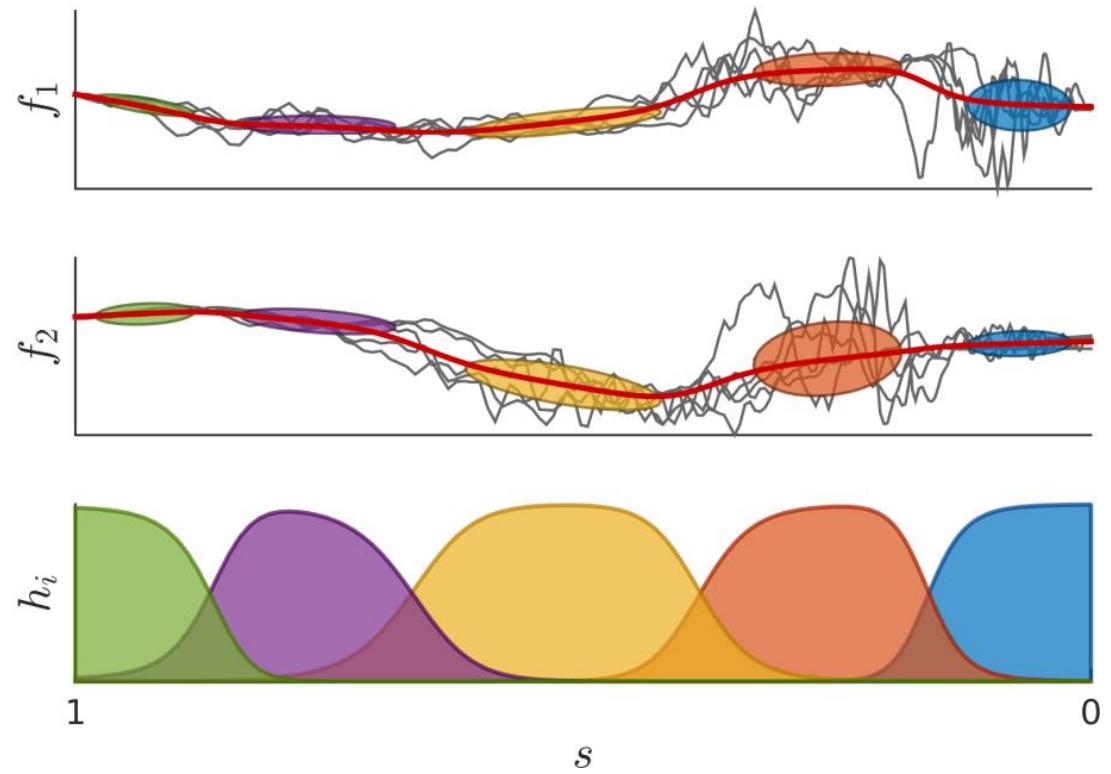
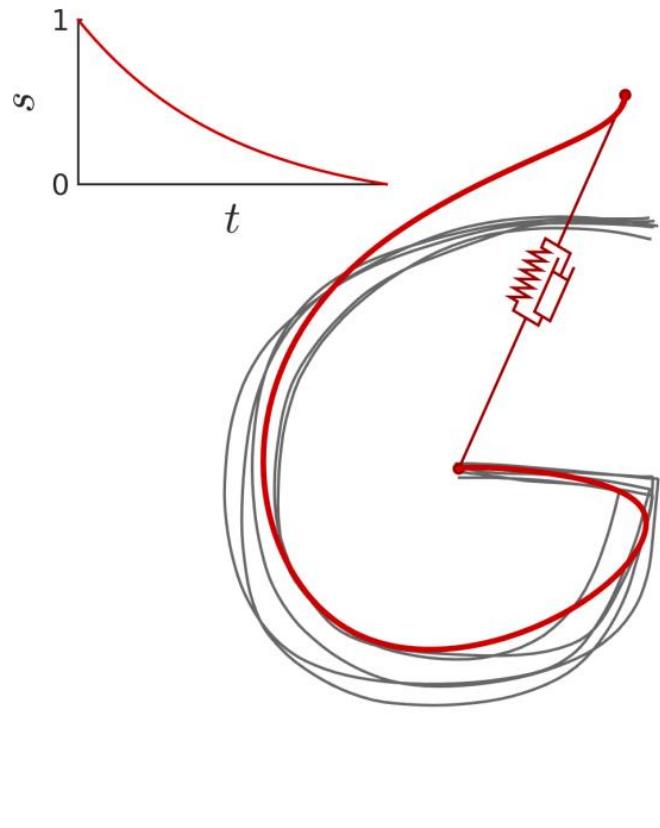
$$\boldsymbol{X}^o = \begin{bmatrix} \ddot{\boldsymbol{x}}_1 - k^p(\boldsymbol{\mu}_T - \boldsymbol{x}_1) + k^\nu \dot{\boldsymbol{x}}_1 \\ \ddot{\boldsymbol{x}}_2 - k^p(\boldsymbol{\mu}_T - \boldsymbol{x}_2) + k^\nu \dot{\boldsymbol{x}}_2 \\ \vdots \\ \ddot{\boldsymbol{x}}_T - k^p(\boldsymbol{\mu}_T - \boldsymbol{x}_T) + k^\nu \dot{\boldsymbol{x}}_T \end{bmatrix} \quad \boldsymbol{X}^i = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_T \end{bmatrix}$$

$$\boldsymbol{W}_k = \text{diag}\left(\phi_k(s_1), \phi_k(s_2), \dots, \phi_k(s_T)\right)$$

$$\hat{\boldsymbol{F}}_k = (\boldsymbol{X}^{i\top} \boldsymbol{W}_k \boldsymbol{X}^i)^{-1} \boldsymbol{X}^{i\top} \boldsymbol{W}_k \boldsymbol{X}^o$$

Dynamical movement primitives with GMR

Learning of $\mathcal{P}(s, \mathbf{x})$ and retrieval of $\mathcal{P}(\mathbf{x}|s)$



Dynamical movement primitives (DMP) - Resources

Softwares

<http://www.idiap.ch/software/pbdlib/>

Matlab codes: demo_DMP01.m

References

[Ijspeert, Nakanishi and Schaal, “*Learning Control Policies For Movement Imitation and Movement recognition*”, NIPS’2003]

[Ijspeert, Nakanishi, Pastor, Hoffmann and Schaal, “*Dynamical movement primitives: Learning attractor models for motor behaviors*”, Neural Computation 25(2), 2013]

[Calinon and Lee, “*Learning Control*”, Humanoid Robotics: a Reference (Springer), 2019]

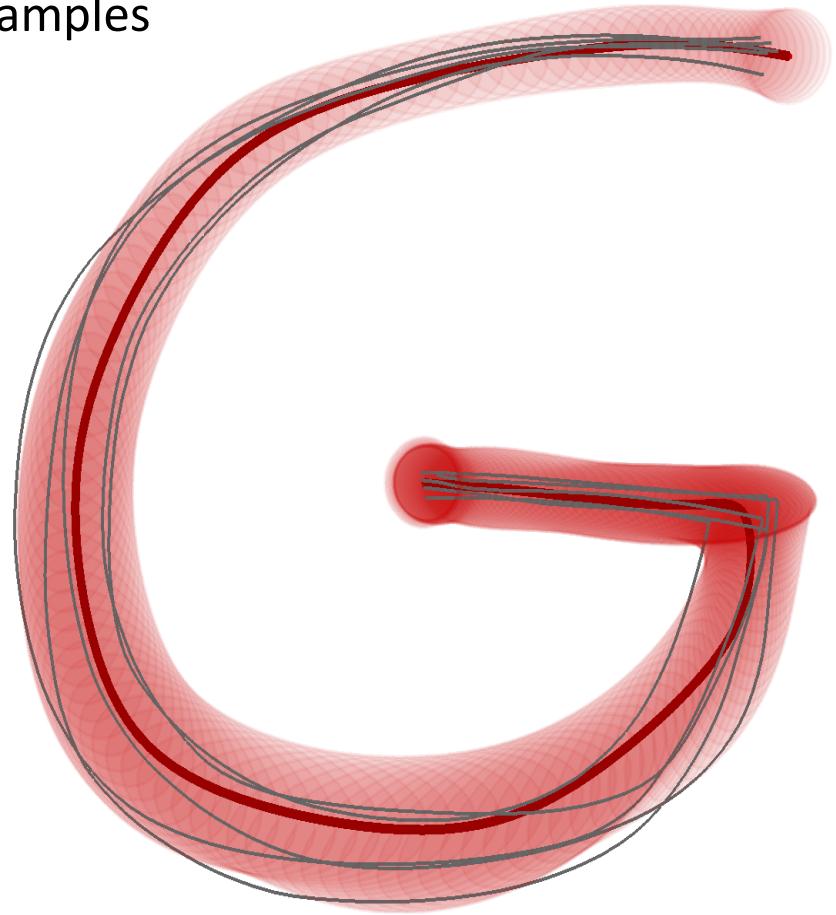
Outline

- **Superposition with basis functions**
 - Bezier curves
 - Locally weighted regression (LWR)
 - Gaussian mixture regression (GMR)
 - Fourier series for periodic motion and ergodic control
- **Dynamical movement primitives (DMP)**
 - Probabilistic movement primitives (ProMP)**
- **Superposition Vs fusion**
 - Product of Gaussians
- **Model predictive control (MPC)**
 - Linear quadratic tracking (LQT)
 - Task-parameterized movement models
- **Differential geometry**
 - Riemannian manifolds

Trajectory distribution

$$\boldsymbol{x}_m = \begin{bmatrix} \boldsymbol{x}_{m,1} \\ \boldsymbol{x}_{m,2} \\ \vdots \\ \boldsymbol{x}_{m,T} \end{bmatrix} \in \mathbb{R}^{DT}$$

M samples



$$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$\frac{1}{M} \sum_{m=1}^M \boldsymbol{x}_m$$

$$\frac{1}{M} \sum_{m=1}^M (\boldsymbol{x}_m - \boldsymbol{\mu})(\boldsymbol{x}_m - \boldsymbol{\mu})^\top$$

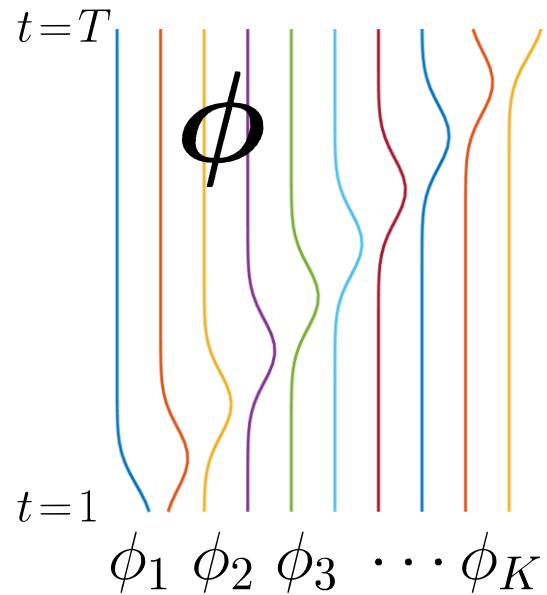
Probabilistic movement primitives (ProMP)

uni-dimensional trajectory:

$$\hat{x} = \sum_{k=1}^K w_k \phi_k$$

$$\hat{x} = \boldsymbol{\phi} \boldsymbol{w}$$

$\boldsymbol{\phi} \in \mathbb{R}^{T \times K}$
 $\boldsymbol{w} \in \mathbb{R}^K$



d-dimensional trajectory:

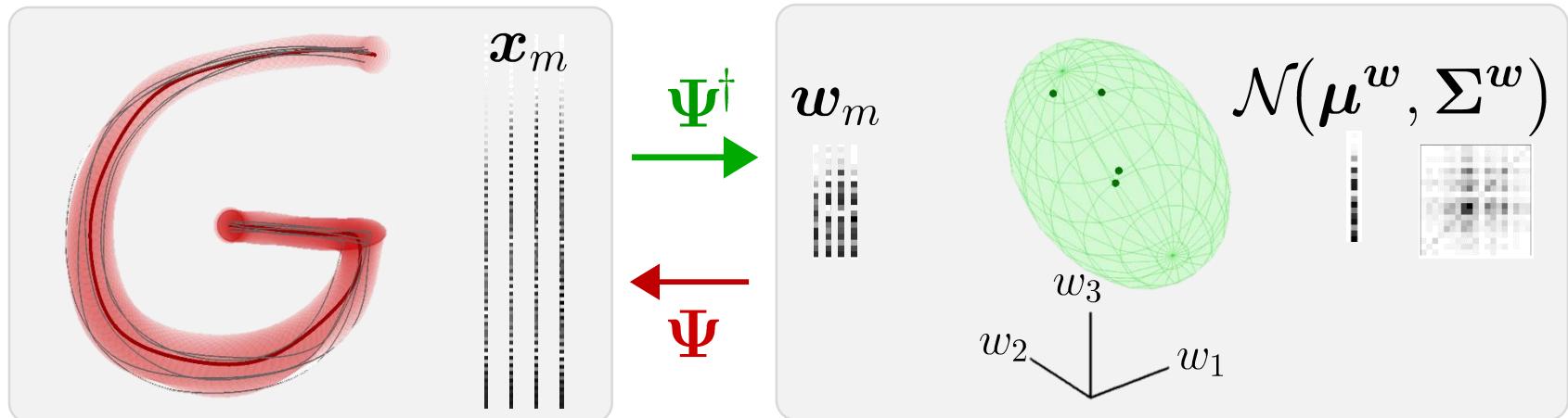
$$\hat{x} = \boldsymbol{\Psi} \boldsymbol{w}$$

$\boldsymbol{\Psi} \in \mathbb{R}^{DT \times DK}$
 $\boldsymbol{w} \in \mathbb{R}^{DK}$

$$\boldsymbol{\Psi} = \begin{bmatrix} \mathbf{I}\phi_1(t_1) & \mathbf{I}\phi_2(t_1) & \cdots & \mathbf{I}\phi_K(t_1) \\ \mathbf{I}\phi_1(t_2) & \mathbf{I}\phi_2(t_2) & \cdots & \mathbf{I}\phi_K(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{I}\phi_1(t_T) & \mathbf{I}\phi_2(t_T) & \cdots & \mathbf{I}\phi_K(t_T) \end{bmatrix}$$

$$\boldsymbol{\Psi} = \boldsymbol{\phi} \otimes \mathbf{I}_d$$

Probabilistic movement primitives (ProMP)



$$\Sigma = \Psi \Sigma^w \Psi^\top$$

$$x \sim \mathcal{N}\left(\Psi \mu^w, \Psi \Sigma^w \Psi^\top + \lambda I\right)$$

$$w_m = (\Psi^\top \Psi)^{-1} \Psi^\top x_m$$

$$x_m = \Psi w_m + \epsilon$$

$$\epsilon \sim \mathcal{N}(0, \lambda I)$$

Probabilistic movement primitives (ProMP)

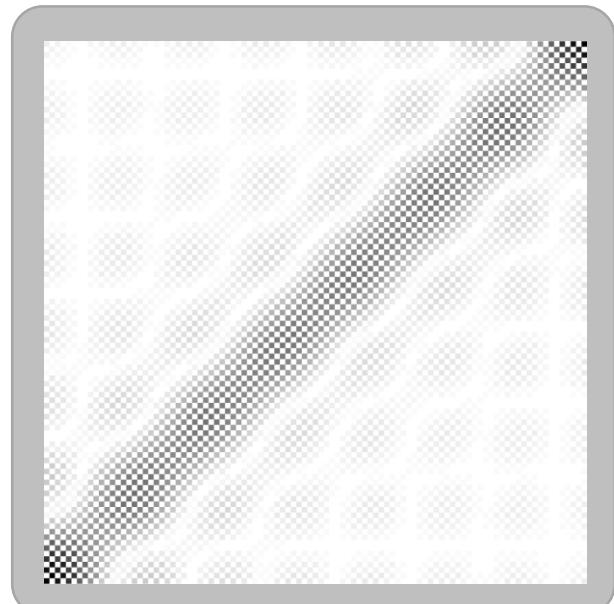
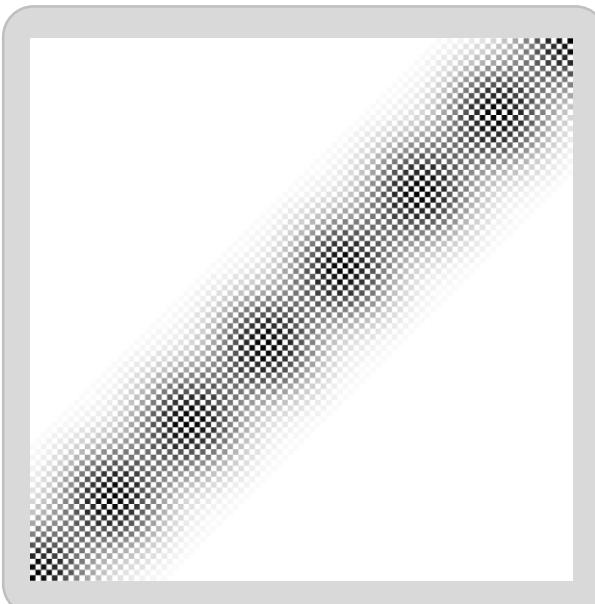
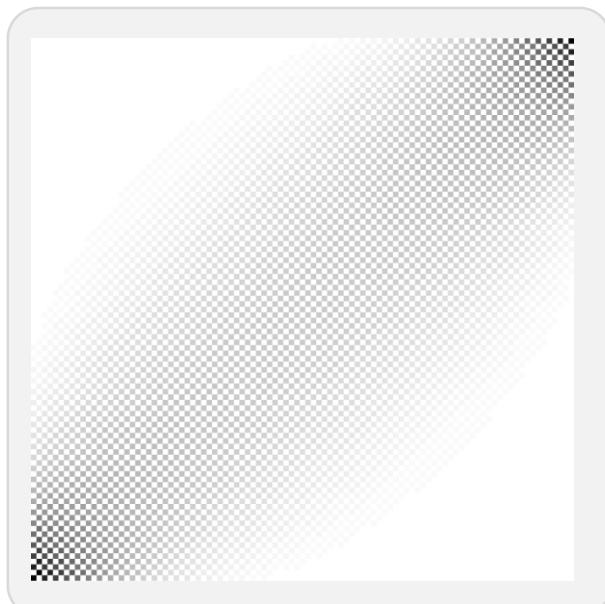
$$\Sigma = \Psi \Sigma^w \Psi^\top$$

$$x \sim \mathcal{N}(\Psi\mu^w, \Psi\Sigma^w\Psi^\top + \lambda I)$$

Bézier curves
(Bernstein bases)

**Locally weighted
regression** (radial bases)

Fourier series
(cosine bases)



Probabilistic movement primitives - Resources

Softwares

<http://www.idiap.ch/software/pbdlib/>

C++ codes: demo_proMP01.cpp

References

[Paraschos, Daniel, Peters and Neumann, “*Probabilistic Movement Primitives*”, NIPS’2013]

[Calinon and Lee, “*Learning Control*”, Humanoid Robotics: a Reference (Springer), 2019]

Outline

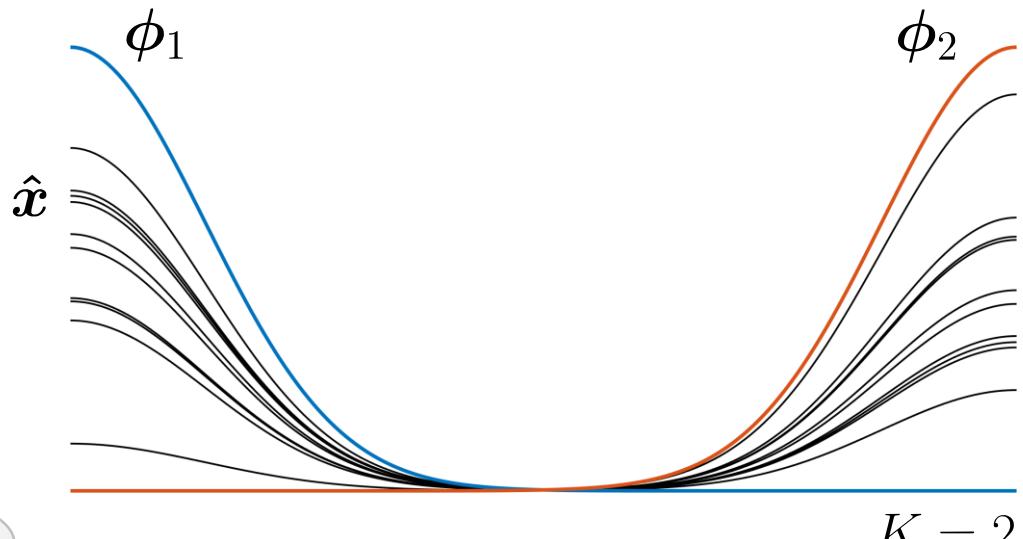
- Superposition with basis functions
 - Bezier curves
 - Locally weighted regression (LWR)
 - Gaussian mixture regression (GMR)
 - Fourier series for periodic motion and ergodic control
- Dynamical movement primitives (DMP)
 - Probabilistic movement primitives (ProMP)
- **Superposition Vs fusion**
 - Product of Gaussians
- Model predictive control (MPC)
 - Linear quadratic tracking (LQT)
 - Task-parameterized movement models
- Differential geometry
 - Riemannian manifolds

Combination of primitives as a fusion problem

Superposition

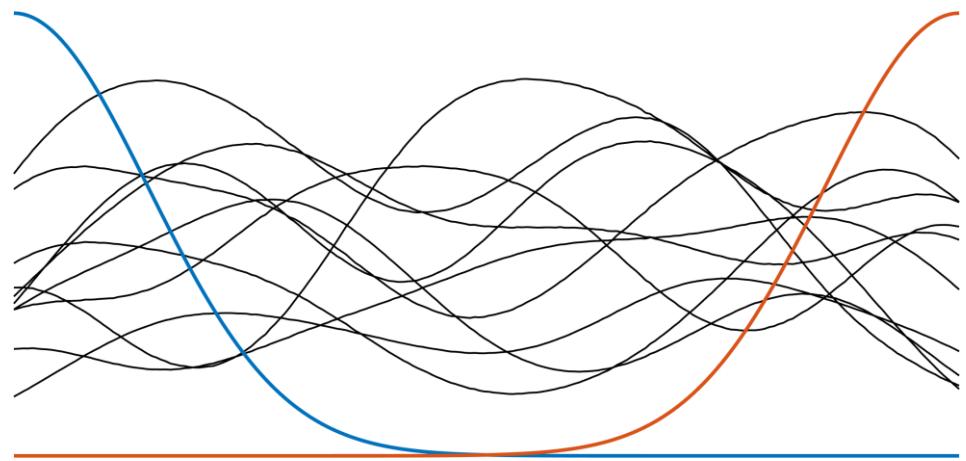
$$\hat{x} = \frac{\sum_{k=1}^K w_k \phi_k}{\sum_{k=1}^K w_k}$$

Choosing between using scalar weights or full weight matrices does not look like a detail!



Fusion

$$\hat{x} = \left(\sum_{k=1}^K W_k \right)^{-1} \sum_{k=1}^K W_k \phi_k$$



Motivating example: A probabilistic view on segment crossing!

$$\begin{aligned}\hat{x} &= \arg \min_x \| \mu_1 - x \|_{W_1}^2 + \| \mu_2 - x \|_{W_2}^2 \\ &= (\mathbf{W}_1 + \mathbf{W}_2)^{-1} (\mathbf{W}_1 \mu_1 + \mathbf{W}_2 \mu_2) \\ &= (\Sigma_1^{-1} + \Sigma_2^{-1})^{-1} (\Sigma_1^{-1} \mu_1 + \Sigma_2^{-1} \mu_2)\end{aligned}$$

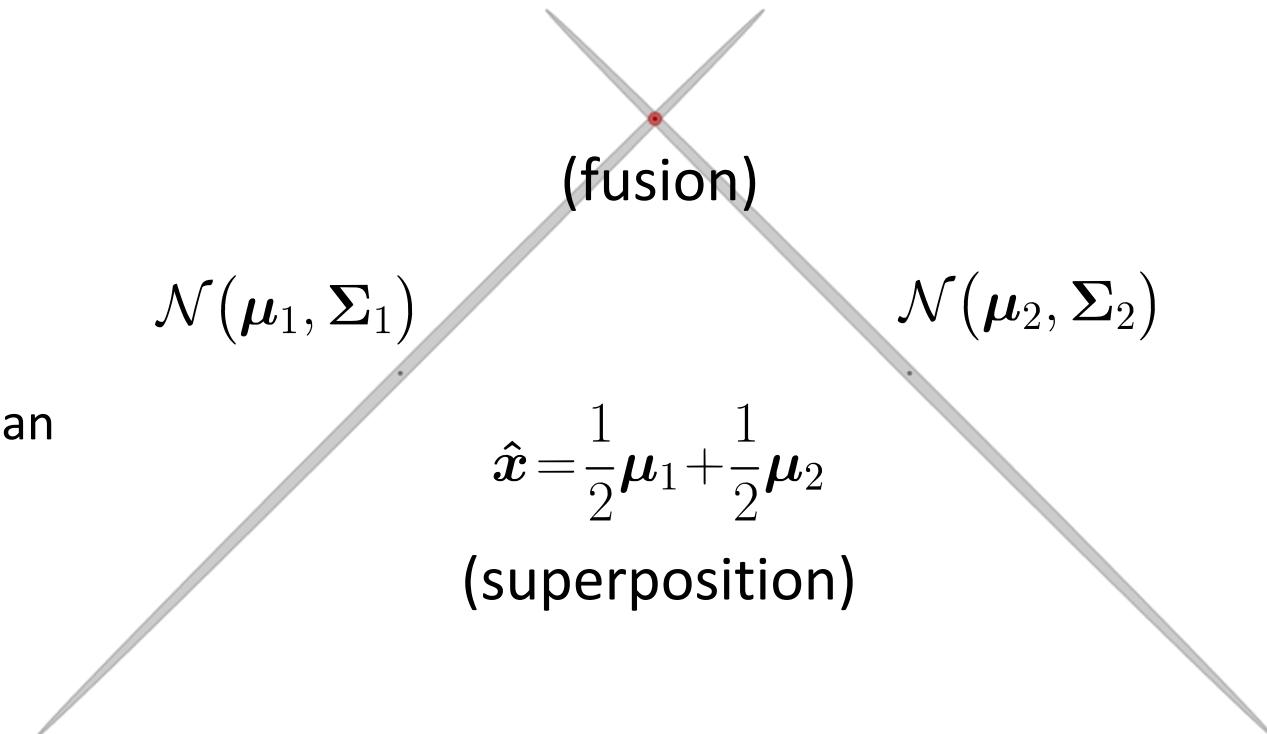
Product of Gaussians

μ_i center of the Gaussian

Σ_i covariance matrix

\mathbf{W}_i precision matrix

$$(\mathbf{W}_i = \Sigma_i^{-1})$$



Kalman filter

Kalman filter with feedback gains

$$\Sigma_t = (\mathbf{I} - \mathbf{K}_t \mathbf{C}) \Sigma_t^{(1)}$$

$$\mu_t = \mu_t^{(1)} + \mathbf{K}_t (\mathbf{y}_t - \mathbf{C} \mu_t^{(1)})$$

$$\mathbf{K}_t = \Sigma_t^{(1)} \mathbf{C}^\top (\Sigma_y + \mathbf{C} \Sigma_t^{(1)} \mathbf{C}^\top)^{-1}$$

$$\mathbf{y}_t = \mathbf{C} \mathbf{x}_t + \mathbf{e}_y$$

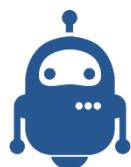
$$\mathbf{e}_y \sim \mathcal{N}(\mathbf{0}, \Sigma_y)$$



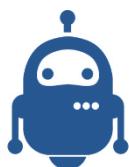
Kalman filter as product of Gaussians

$$\Sigma_t = \left(\Sigma_t^{(1)-1} + \Sigma_t^{(2)-1} \right)^{-1}$$

$$\mu_t = \Sigma_t \left(\Sigma_t^{(1)-1} \mu_t^{(1)} + \Sigma_t^{(2)-1} \mu_t^{(2)} \right)$$



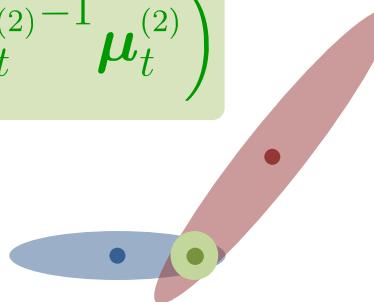
t=0



t=1



t=2



$$\mu_t^{(2)} \triangleq \mathbf{C}^\dagger \mathbf{y}_t$$

$$\Sigma_t^{(2)} \triangleq \mathbf{C}^\dagger \Sigma_y \mathbf{C}^{\dagger \top}$$

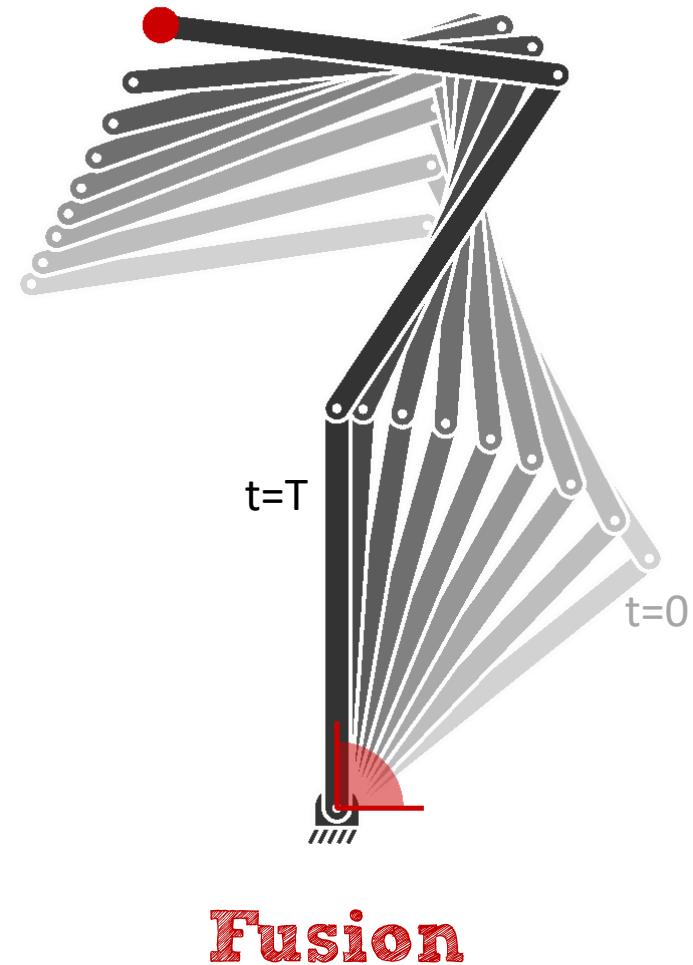
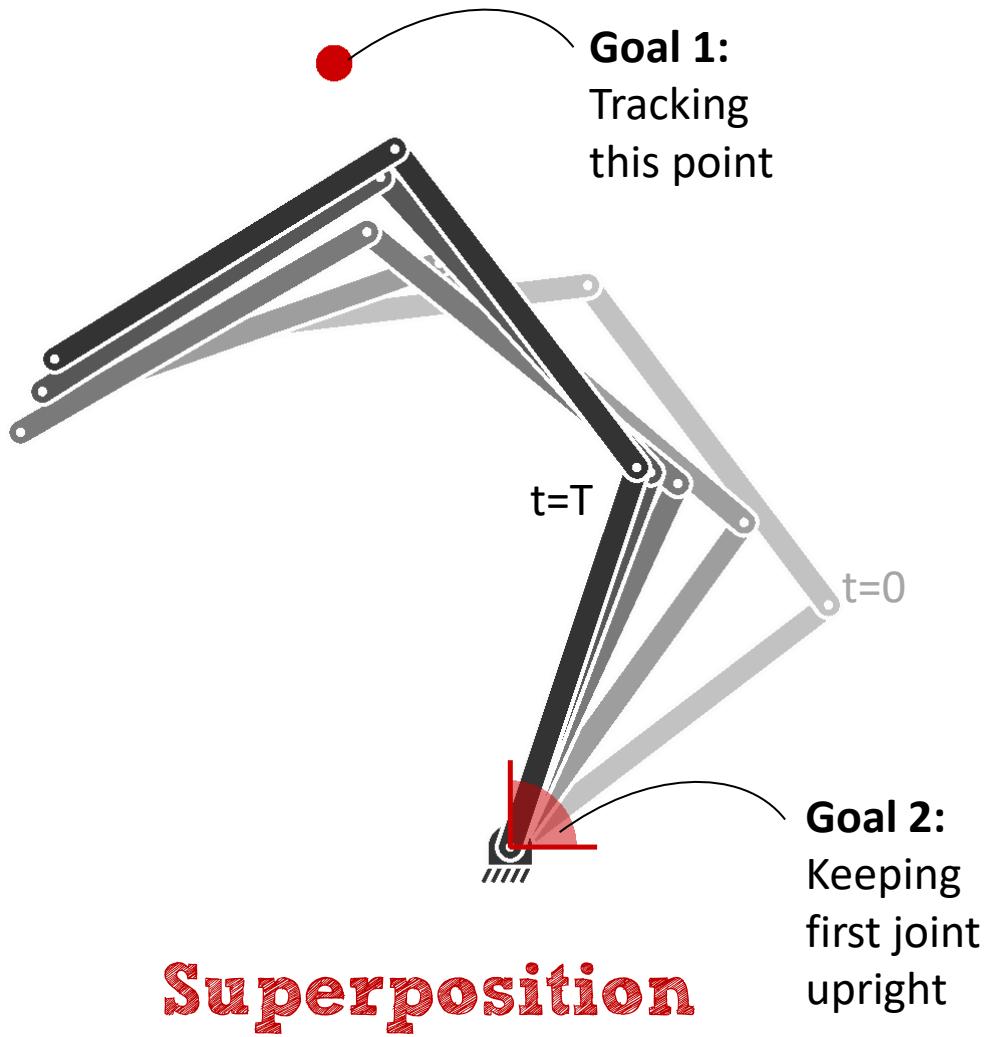
$$\mathbf{x}_t = \mathbf{A} \mathbf{x}_{t-1} + \mathbf{B} \mathbf{u}_t + \mathbf{e}_x$$

$$\mathbf{e}_x \sim \mathcal{N}(\mathbf{0}, \Sigma_x)$$

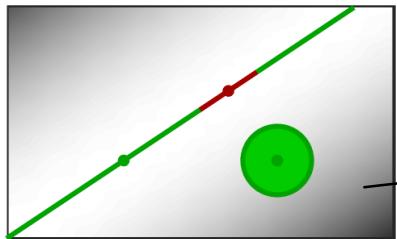
$$\mu_t^{(1)} \triangleq \mathbf{A} \mathbf{x}_{t-1} + \mathbf{B} \mathbf{u}_t$$

$$\Sigma_t^{(1)} \triangleq \mathbf{A} \Sigma_{t-1} \mathbf{A}^\top + \Sigma_x$$

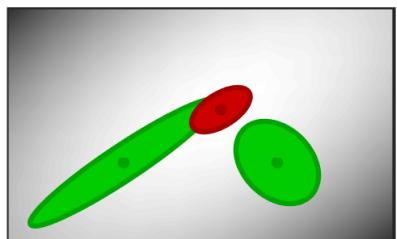
Motivating example: Fusion of IK and joint angle controllers



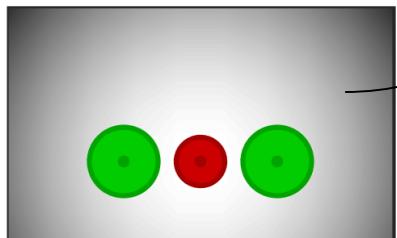
Combination of primitives as a fusion problem



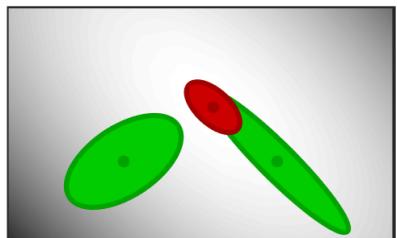
$$\mathcal{N}(\mu, \Sigma) \propto \mathcal{N}(\mu^{(1)}, \Sigma^{(1)}) \mathcal{N}(\mu^{(2)}, \Sigma^{(2)})$$



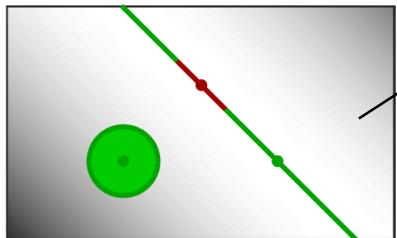
Scalar superposition



Null space projection
(hierarchy constraints)



The full weight matrices approach
covers both **scalar weights**
(with isotropic diagonal matrix)
and **null space projection**
operations!



Outline

- Superposition with basis functions
 - Bezier curves
 - Locally weighted regression (LWR)
 - Gaussian mixture regression (GMR)
 - Fourier series for periodic motion and ergodic control
- Dynamical movement primitives (DMP)
 - Probabilistic movement primitives (ProMP)
- Superposition Vs fusion
 - Product of Gaussians
- Model predictive control (MPC)
 - Linear quadratic tracking (LQT)
 - Task-parameterized movement models
- Differential geometry
 - Riemannian manifolds

Linear quadratic tracking (LQT)

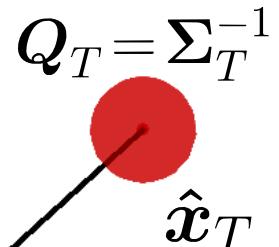
$$\min_{\boldsymbol{u}} \sum_{t=1}^T \left\| \boldsymbol{\mu}_t - \boldsymbol{x}_t \right\|_{Q_t}^2 + \left\| \boldsymbol{u}_t \right\|_{R_t}^2$$

Track path!

Use low control commands!

s.t. $\boldsymbol{x}_{t+1} = \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t$

System dynamics

$$Q_T = \Sigma_T^{-1}$$


Model predictive control (MPC):

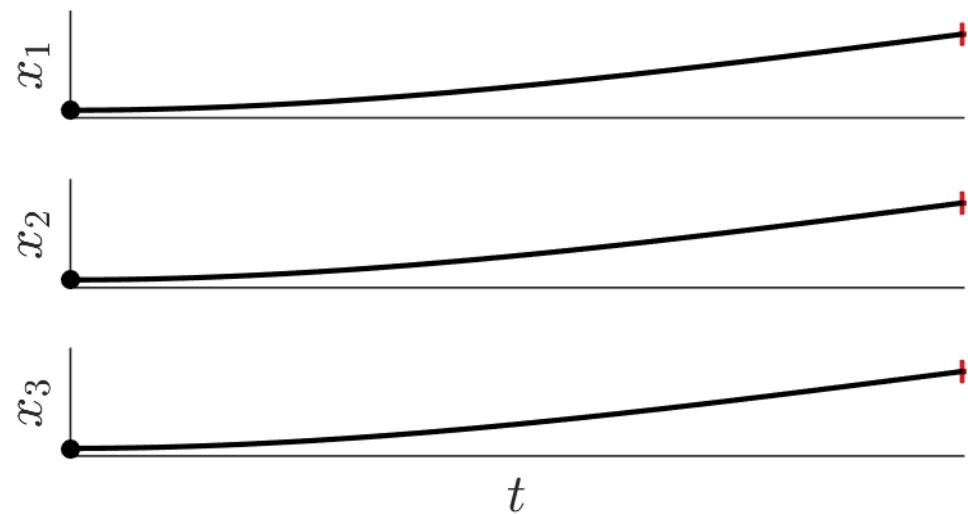
\boldsymbol{x}_t state variable (position+velocity)

$\boldsymbol{\mu}_t$ desired state

\boldsymbol{u}_t control command (acceleration)

Q_t precision matrix

R_t control weight matrix



How to solve this objective function?

$$\min_{\boldsymbol{u}} \sum_{t=1}^T \left\| \boldsymbol{\mu}_t - \boldsymbol{x}_t \right\|_{Q_t}^2 + \left\| \boldsymbol{u}_t \right\|_{R_t}^2$$

Track path!

Use low control commands!

s.t. $\boldsymbol{x}_{t+1} = \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t$

System dynamics

**Pontryagin's
maximum principle**
→ Riccati equation

(the Physicist perspective)



**Dynamic
programming**
*(the Computer
Scientist perspective)*



Linear algebra
*(the Algebraist
perspective)*



Let's first re-organize the objective function...

$$\begin{aligned} c &= \sum_{t=1}^T \left((\mu_t - x_t)^\top Q_t (\mu_t - x_t) + u_t^\top R_t u_t \right) \\ &= (\mu - x)^\top Q (\mu - x) + u^\top R u \end{aligned}$$



$$Q = \begin{bmatrix} Q_1 & 0 & \cdots & 0 \\ 0 & Q_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Q_T \end{bmatrix}$$

$$R = \begin{bmatrix} R_1 & 0 & \cdots & 0 \\ 0 & R_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_T \end{bmatrix}$$

$$\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_T \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_T \end{bmatrix}$$

$$u = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_T \end{bmatrix}$$

Let's then re-organize the constraint...

$$\mathbf{x}_{t+1} = \mathbf{A} \mathbf{x}_t + \mathbf{B} \mathbf{u}_t$$



$$\mathbf{x}_2 = \mathbf{A}\mathbf{x}_1 + \mathbf{B}\mathbf{u}_1$$

$$\mathbf{x}_3 = \mathbf{A}\mathbf{x}_2 + \mathbf{B}\mathbf{u}_2 = \mathbf{A}(\mathbf{A}\mathbf{x}_1 + \mathbf{B}\mathbf{u}_1) + \mathbf{B}\mathbf{u}_2$$

$$\vdots$$

$$\mathbf{x}_T = \mathbf{A}^{T-1}\mathbf{x}_1 + \mathbf{A}^{T-2}\mathbf{B}\mathbf{u}_1 + \mathbf{A}^{T-3}\mathbf{B}\mathbf{u}_2 + \cdots + \mathbf{B}_{T-1}\mathbf{u}_{T-1}$$

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \vdots \\ \mathbf{x}_T \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{I} \\ \mathbf{A} \\ \mathbf{A}^2 \\ \vdots \\ \mathbf{A}^{T-1} \end{bmatrix}}_{\mathbf{S}^x} \mathbf{x}_1 + \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{B} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{AB} & \mathbf{B} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{A}^{T-2}\mathbf{B} & \mathbf{A}^{T-3}\mathbf{B} & \cdots & \mathbf{B} & \mathbf{0} \end{bmatrix}}_{\mathbf{S}^u} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_T \end{bmatrix}$$

$$\mathbf{x} = \mathbf{S}^x \mathbf{x}_1 + \mathbf{S}^u \mathbf{u}$$

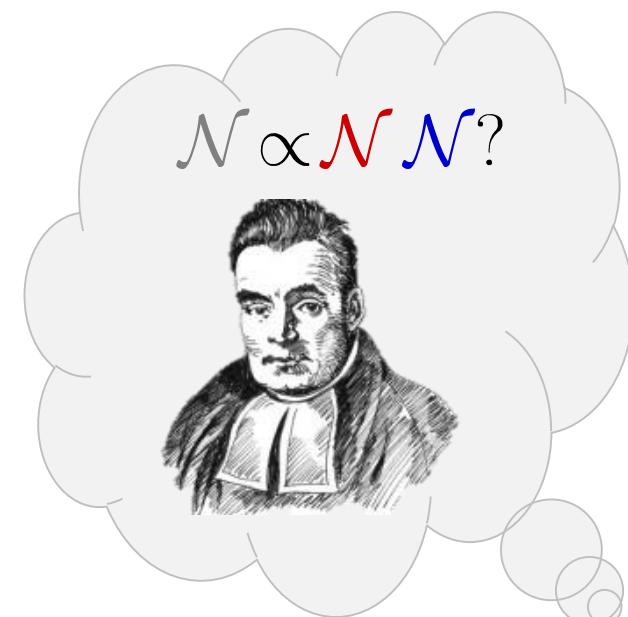
Linear quadratic tracking: Analytic solution

The constraint can then be put into the objective function:

$$\begin{aligned}x &= S^x x_1 + S^u u \\c &= (\mu - x)^\top Q (\mu - x) + u^\top R u \\&= (\mu - S^x x_1 - S^u u)^\top Q (\mu - S^x x_1 - S^u u) + u^\top R u\end{aligned}$$

Solving for u results in the analytic solution:

$$\hat{u} = (S^{u^\top} Q S^u + R)^{-1} S^{u^\top} Q (\mu - S^x x_1)$$



MPC/LQT as a product of Gaussians

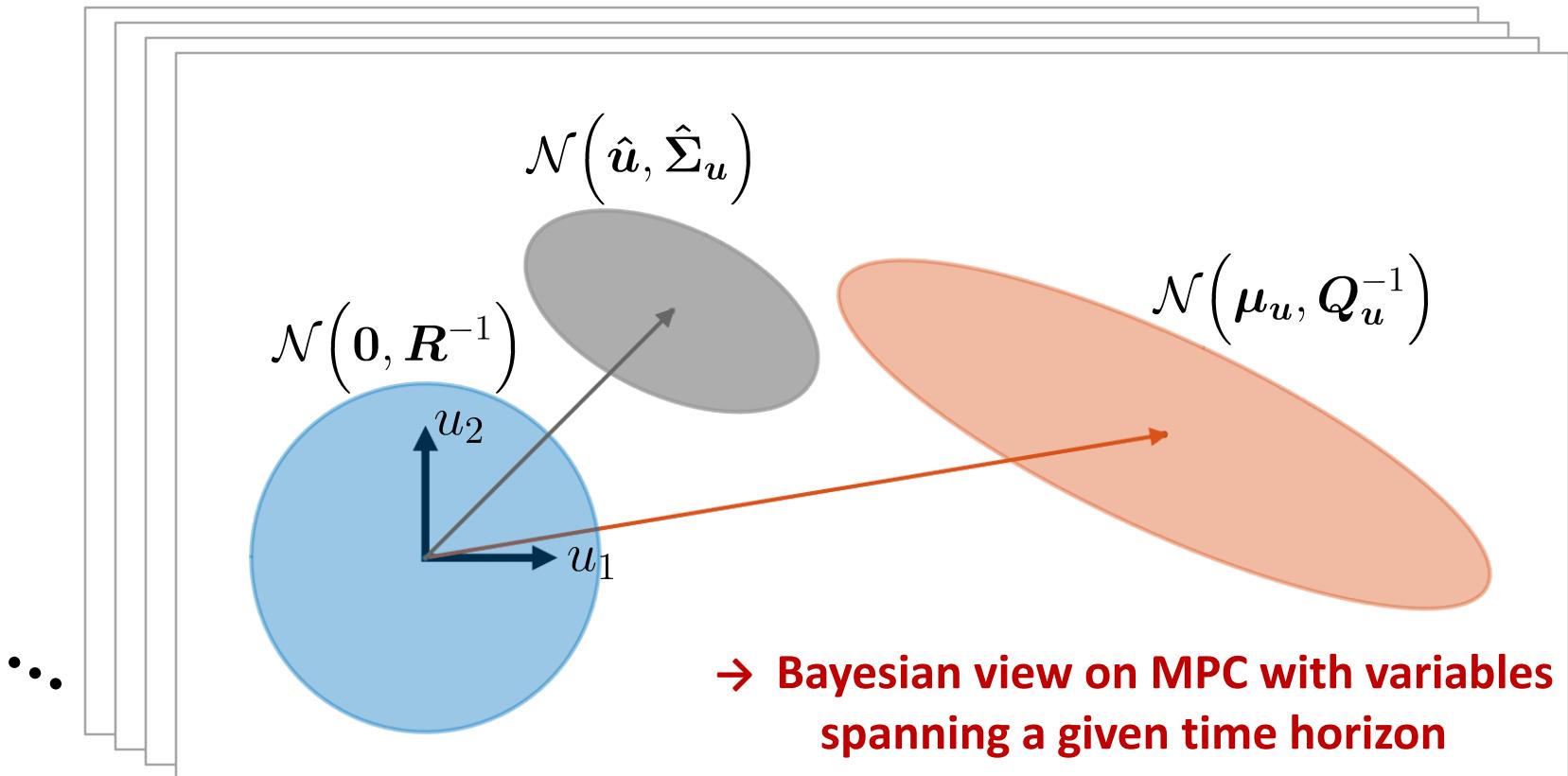
$$\min_u \sum_{t=1}^T \left\| \boldsymbol{\mu}_t - \boldsymbol{x}_t \right\|_{Q_t}^2 + \left\| \boldsymbol{u}_t \right\|_{R_t}^2$$

Track path!
 Use low control commands!

$$\mathcal{N}\left(\hat{\boldsymbol{u}}, \hat{\Sigma}^u\right) \propto \mathcal{N}\left(\boldsymbol{\mu}_u, Q_u^{-1}\right) \mathcal{N}\left(0, R^{-1}\right)$$

$$\boldsymbol{\mu}_u \triangleq S^{u\dagger}(\boldsymbol{\mu} - S^x \boldsymbol{x}_1)$$

$$Q_u \triangleq S^{u\top} Q S^u$$



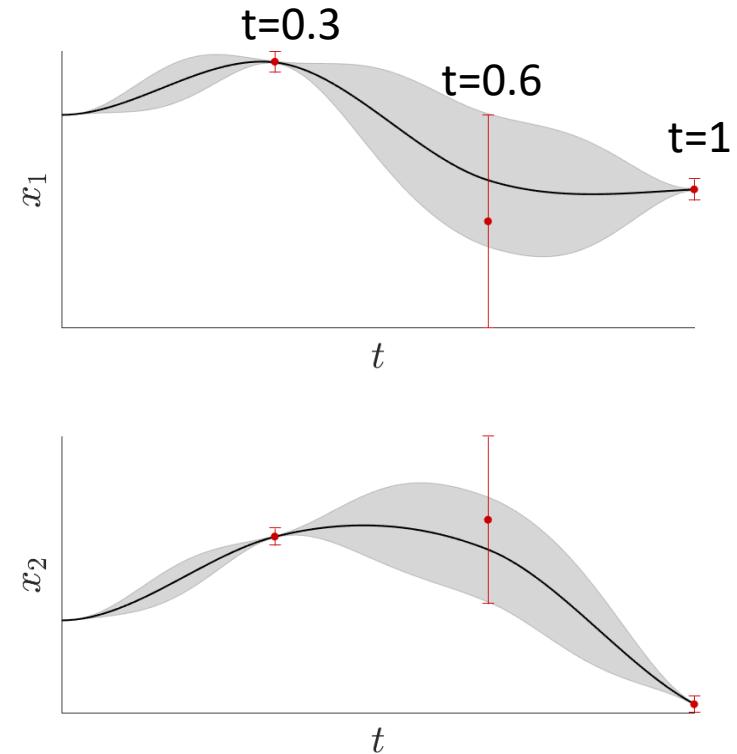
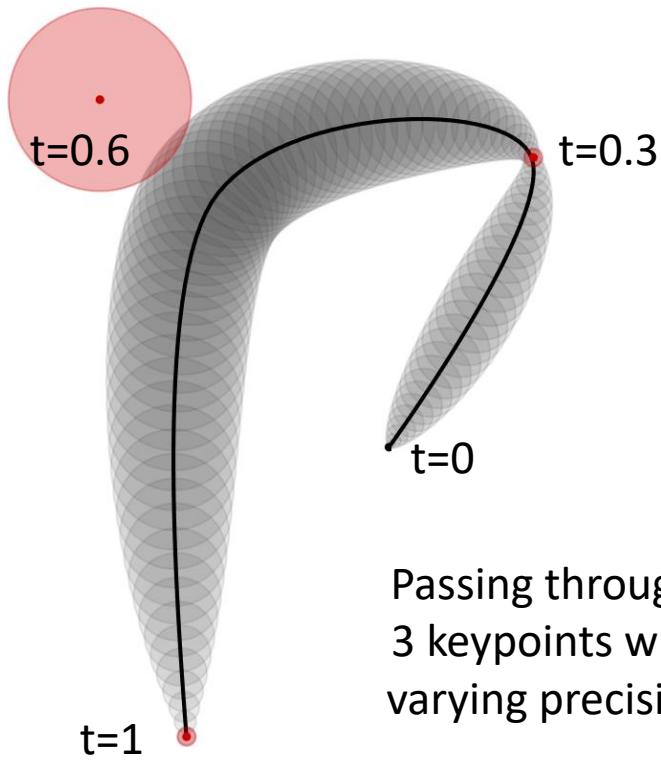
Probabilistic representation of MPC/LQT

$$\hat{u} = (S^{u^\top} Q S^u + R)^{-1} S^{u^\top} Q (\mu - S^x x_1)$$
$$\hat{\Sigma}^u = (S^{u^\top} Q S^u + R)^{-1}$$



$$\hat{x} = S^x x_1 + S^u \hat{u}$$
$$\hat{\Sigma}^x = S^u (S^{u^\top} Q S^u + R)^{-1} S^{u^\top}$$

The distribution in control space can
be projected back to the state space

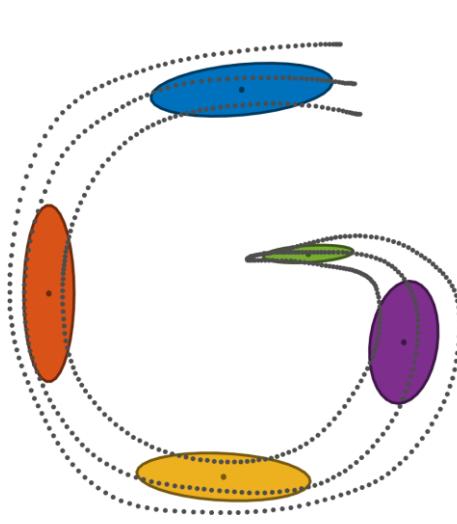


Model Predictive Control (MPC) combined with probabilistic representation of movement primitives

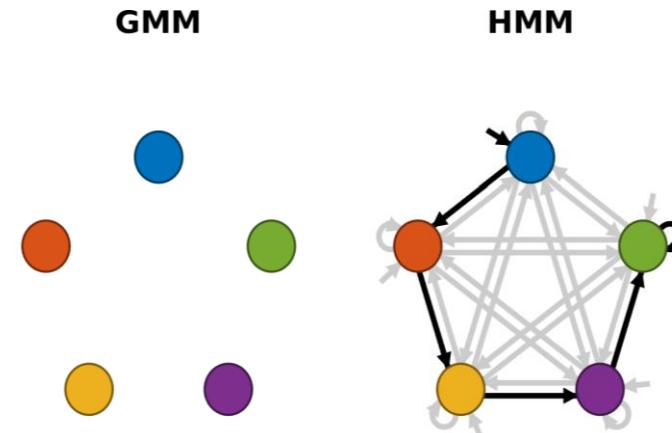


$$c = (\boldsymbol{\mu} - \boldsymbol{x})^\top \mathbf{Q} (\boldsymbol{\mu} - \boldsymbol{x}) + \boldsymbol{u}^\top \mathbf{R} \boldsymbol{u}$$
$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \\ \vdots \\ \boldsymbol{\mu}_T \end{bmatrix} \quad \mathbf{Q} = \begin{bmatrix} \mathbf{Q}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{Q}_T \end{bmatrix}$$

Hidden semi-Markov model (HSMM)

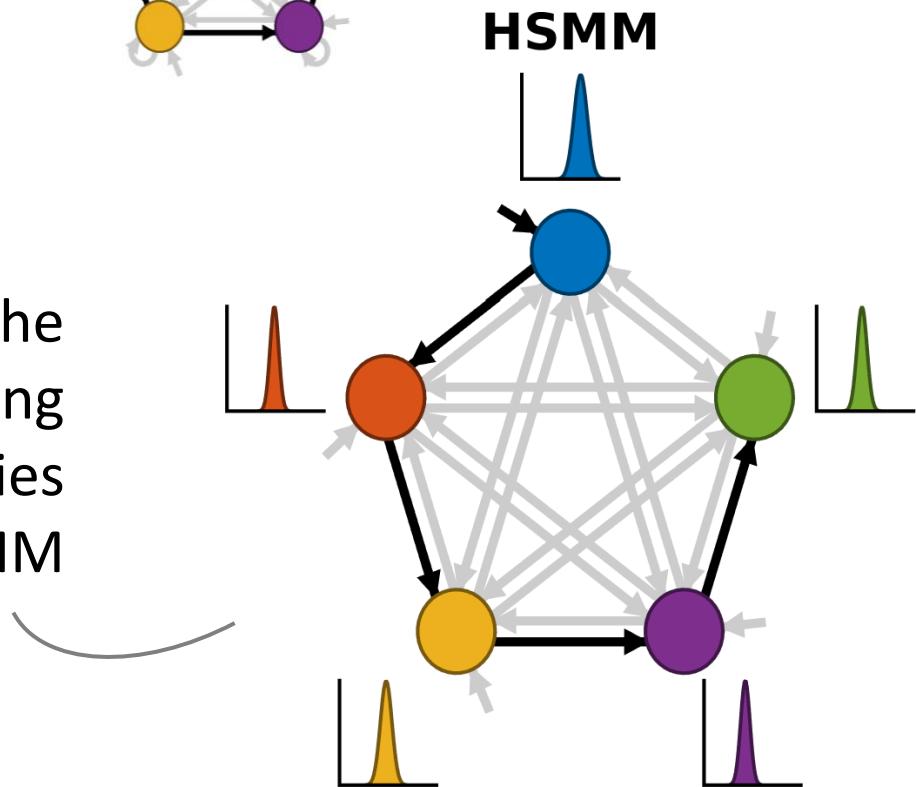


GMM



HMM

HSMM provides a model of the state duration instead of relying on self-transition probabilities as in standard HMM



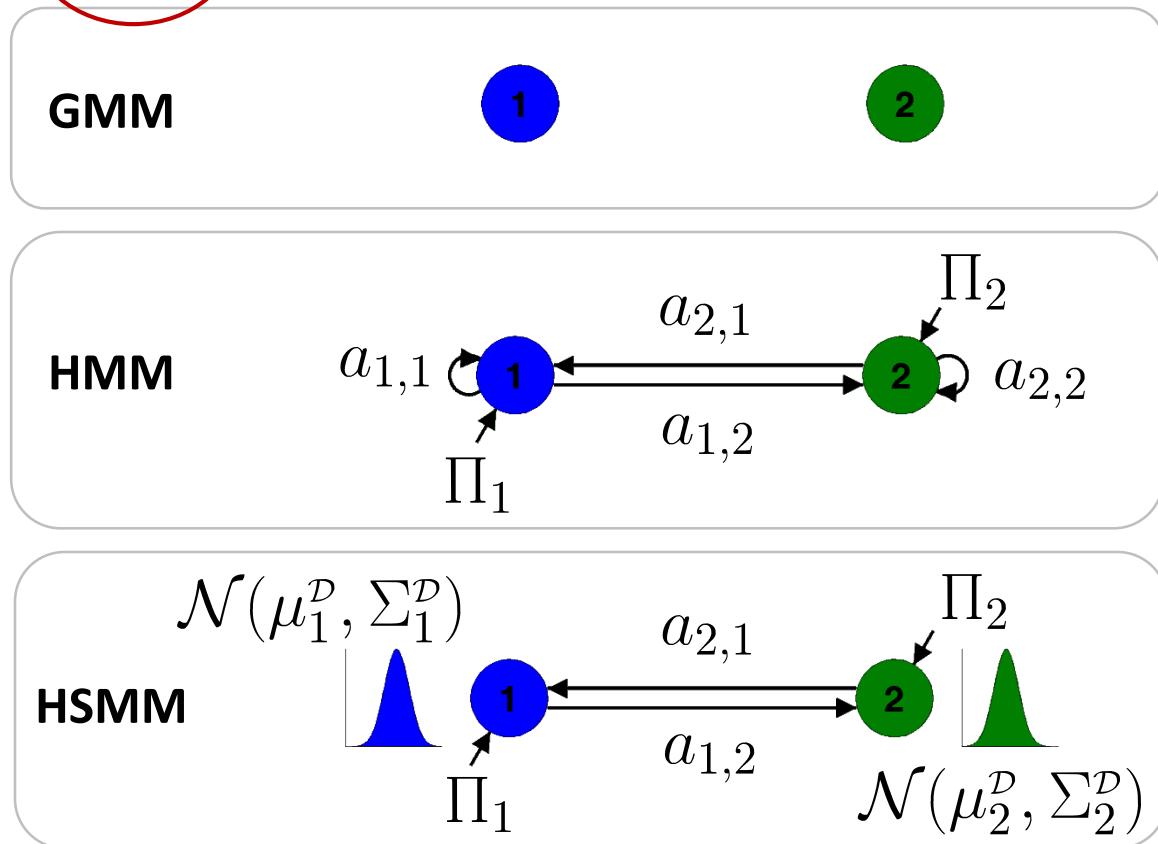
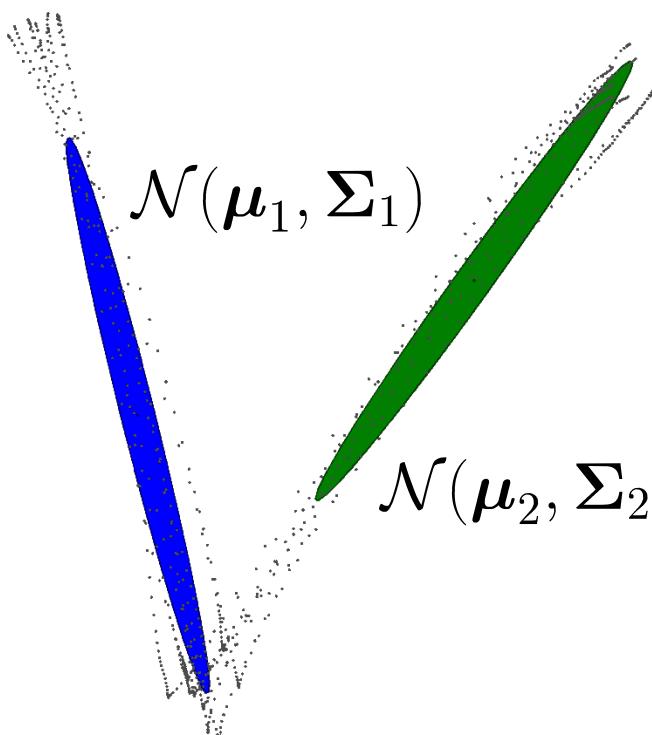
Hidden semi-Markov model (HSMM)

$$\Theta^{\text{GMM}} = \{\pi_i, \mu_i, \Sigma_i\}_{i=1}^K$$

$$\Theta^{\text{HMM}} = \{\{a_{i,j}\}_{j=1}^K, \Pi_i, \mu_i, \Sigma_i\}_{i=1}^K$$

$$\Theta^{\text{HSMM}} = \{\{a_{i,j}\}_{j=1, j \neq i}^K, \Pi_i, \mu_i^{\mathcal{D}}, \Sigma_i^{\mathcal{D}}, \mu_i, \Sigma_i\}_{i=1}^K$$

Parametric duration distribution



Learning minimal intervention controllers

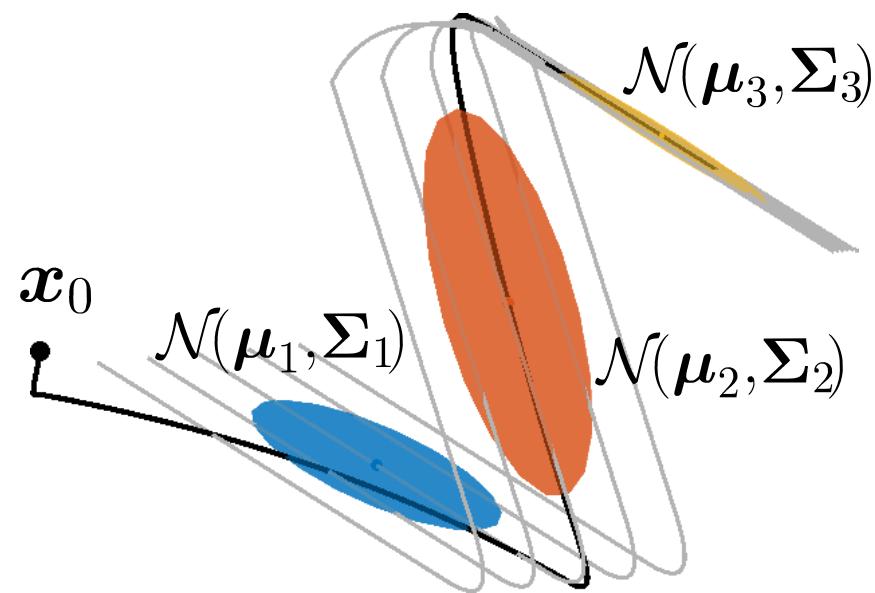
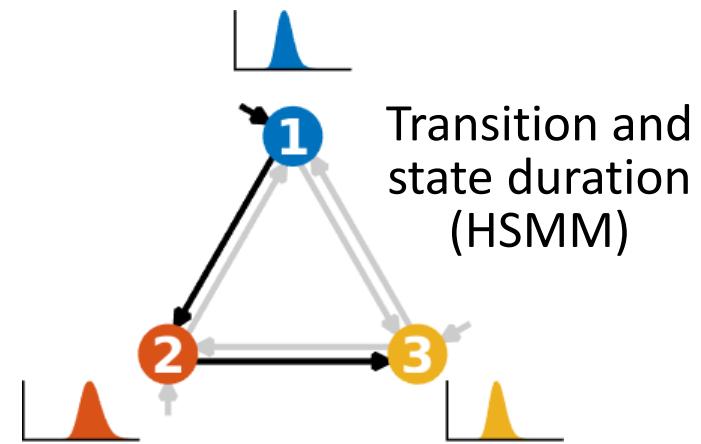
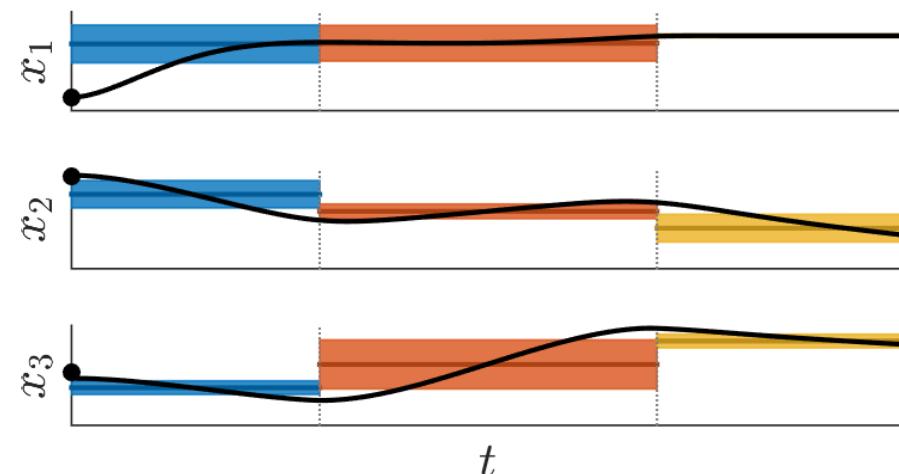
$$\min_u \sum_{t=1}^T \|\hat{x}_t - x_t\|_{Q_t}^2 + \|u_t\|_{R_t}^2$$

s.t. $\dot{x}_t = Ax_t + Bu_t$

Stepwise reference path given by:

$$\hat{x}_t = \mu_{s_t} \quad Q_t = \Sigma_{s_t}^{-1}$$

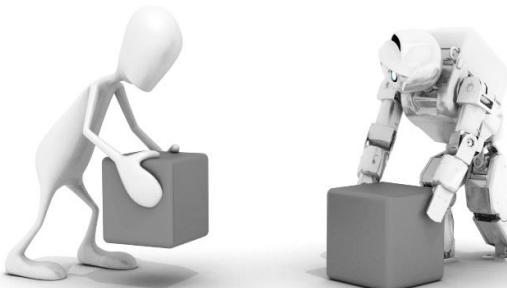
s_t 111111111122222222222333333333



μ_i center of the Gaussian
 Σ_i covariance matrix

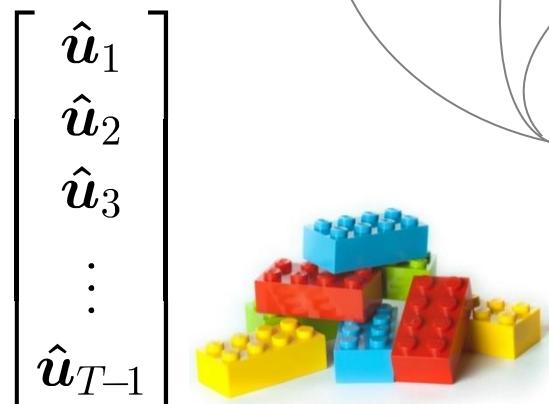
Learning minimal intervention controllers

$$s = \{1, 1, 1, 2, \dots, 4\}$$



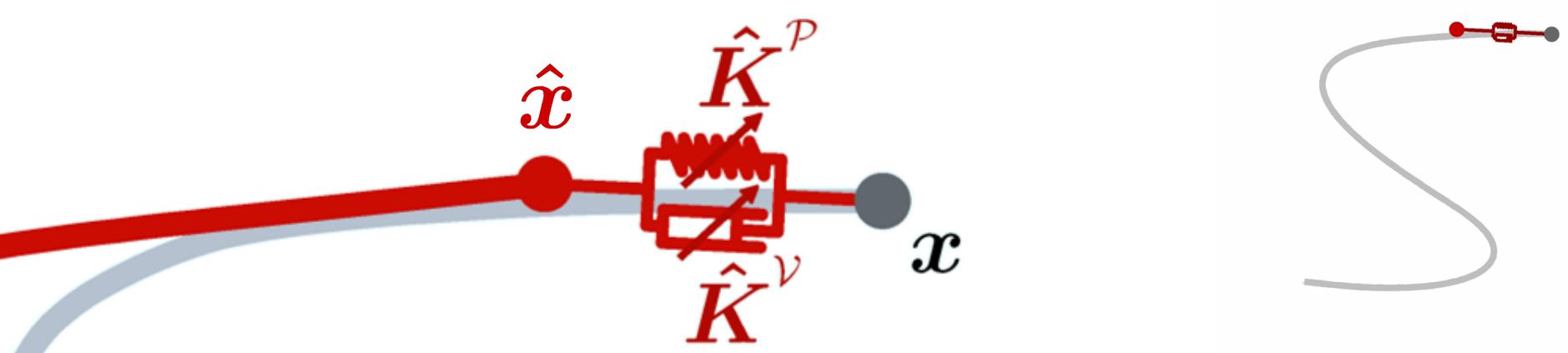
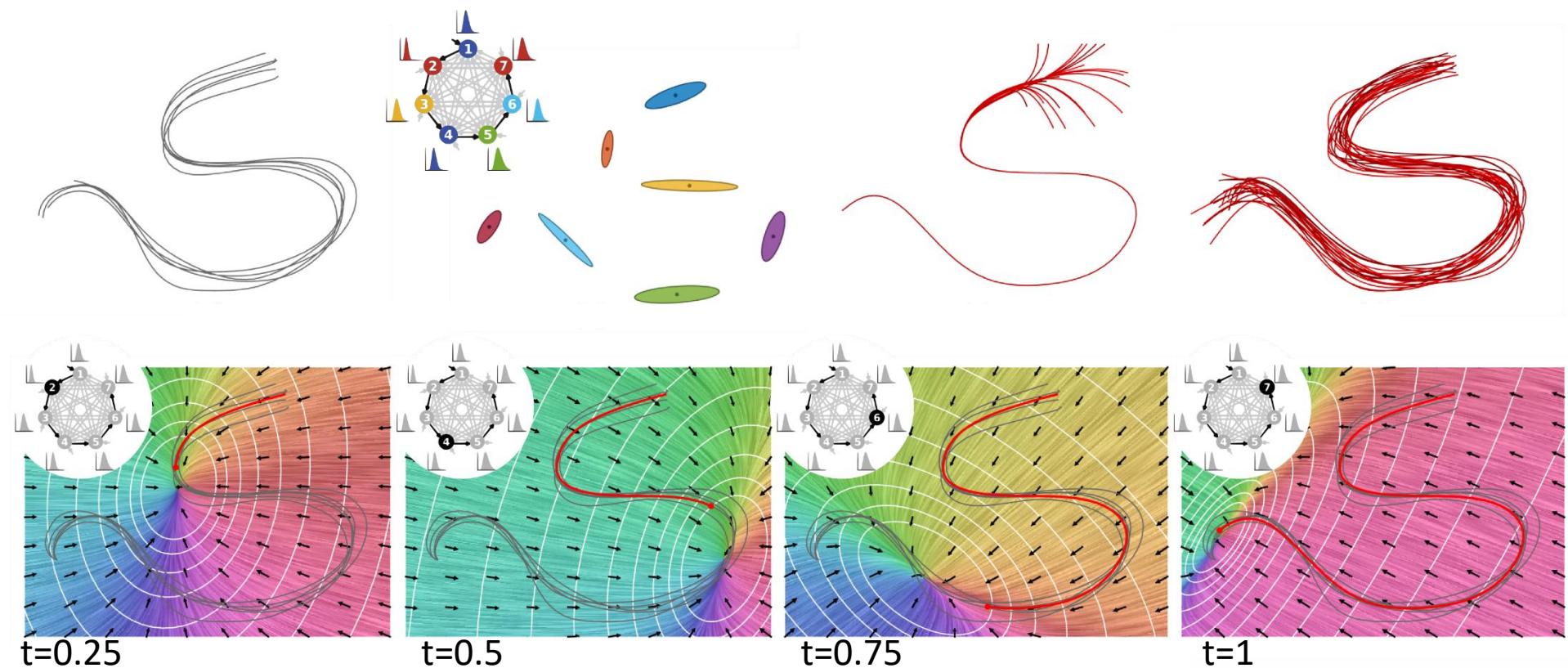
$$\begin{bmatrix} \Sigma_1^{-1} & 0 & 0 & 0 & \cdots & 0 \\ 0 & \Sigma_1^{-1} & 0 & 0 & \cdots & 0 \\ 0 & 0 & \Sigma_1^{-1} & 0 & \cdots & 0 \\ 0 & 0 & 0 & \Sigma_2^{-1} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \Sigma_K^{-1} \end{bmatrix} \begin{bmatrix} \mu_1 \\ \mu_1 \\ \mu_1 \\ \mu_2 \\ \vdots \\ \mu_K \end{bmatrix}$$

$$\hat{u} = (S^{u^\top} Q S^u + R)^{-1} S^{u^\top} Q (\mu - S^x x_1)$$



$$\begin{bmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \hat{u}_3 \\ \vdots \\ \hat{u}_{T-1} \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & \cdots & 0 \\ B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{T-2}B & A^{T-3}B & \cdots & B \end{bmatrix} \quad \begin{bmatrix} I \\ A \\ A^2 \\ \vdots \\ A^{T-1} \end{bmatrix}$$

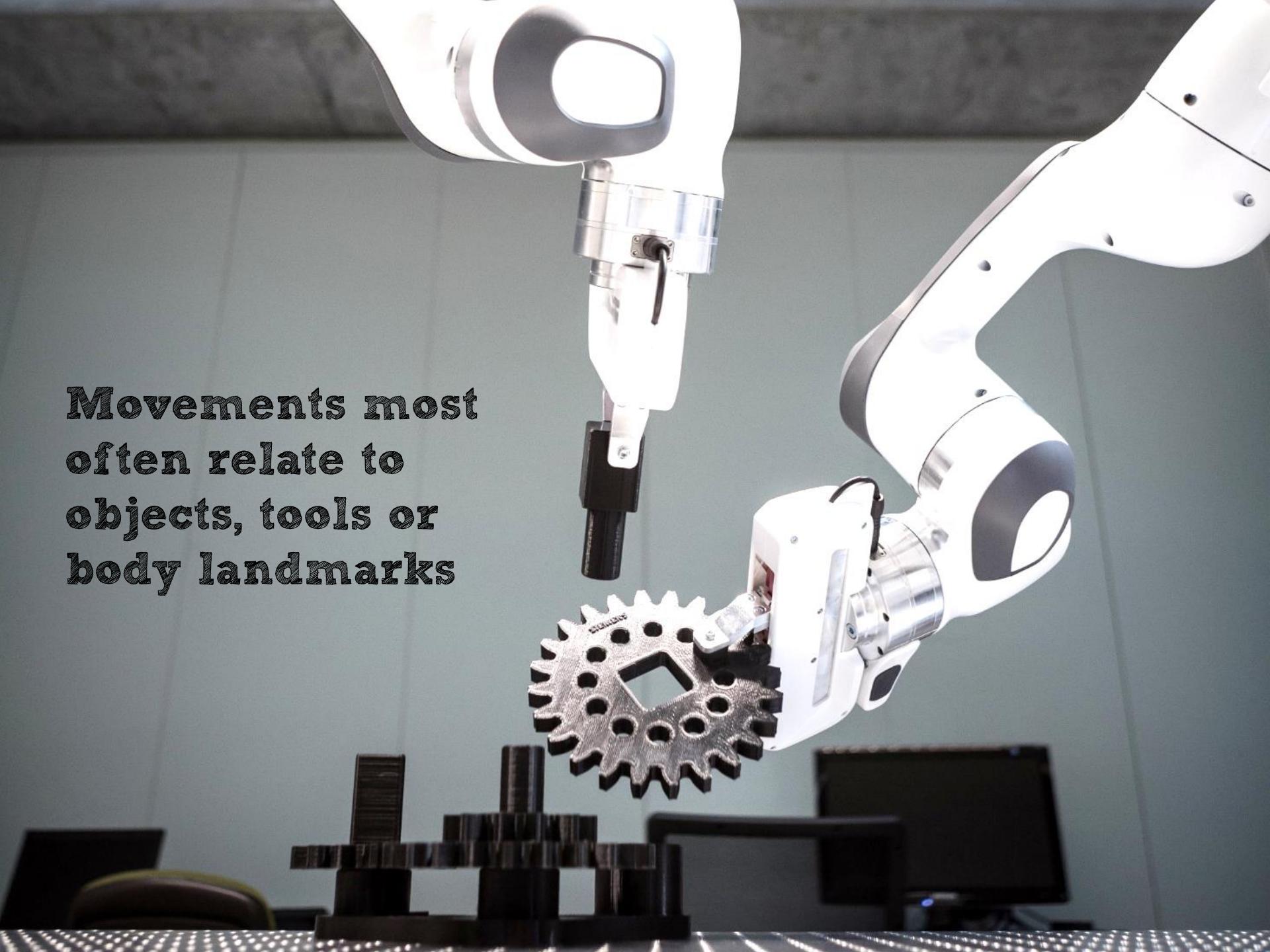
Learning controllers instead of trajectories



Outline

- **Superposition with basis functions**
 - Bezier curves
 - Locally weighted regression (LWR)
 - Gaussian mixture regression (GMR)
 - Fourier series for periodic motion and ergodic control
- **Dynamical movement primitives (DMP)**
 - Probabilistic movement primitives (ProMP)
- **Superposition Vs fusion**
 - Product of Gaussians
- **Model predictive control (MPC)**
 - Linear quadratic tracking (LQT)
 - Task-parameterized movement models**
- **Differential geometry**
 - Riemannian manifolds

Movements most often relate to objects, tools or body landmarks

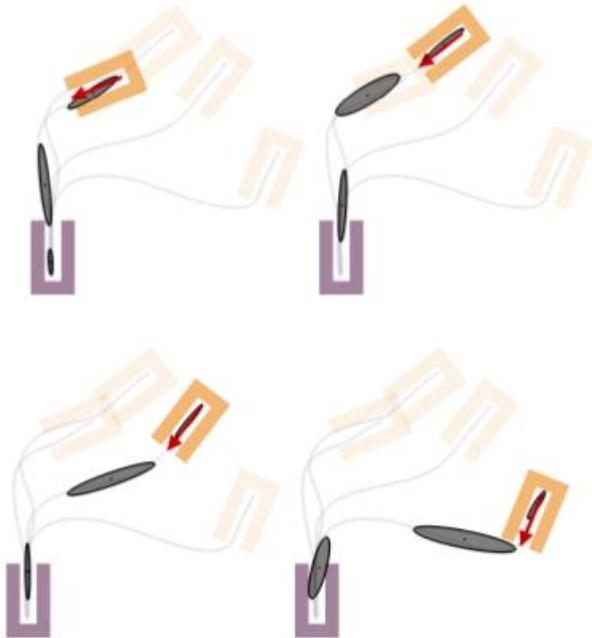


Conditioning-based approach

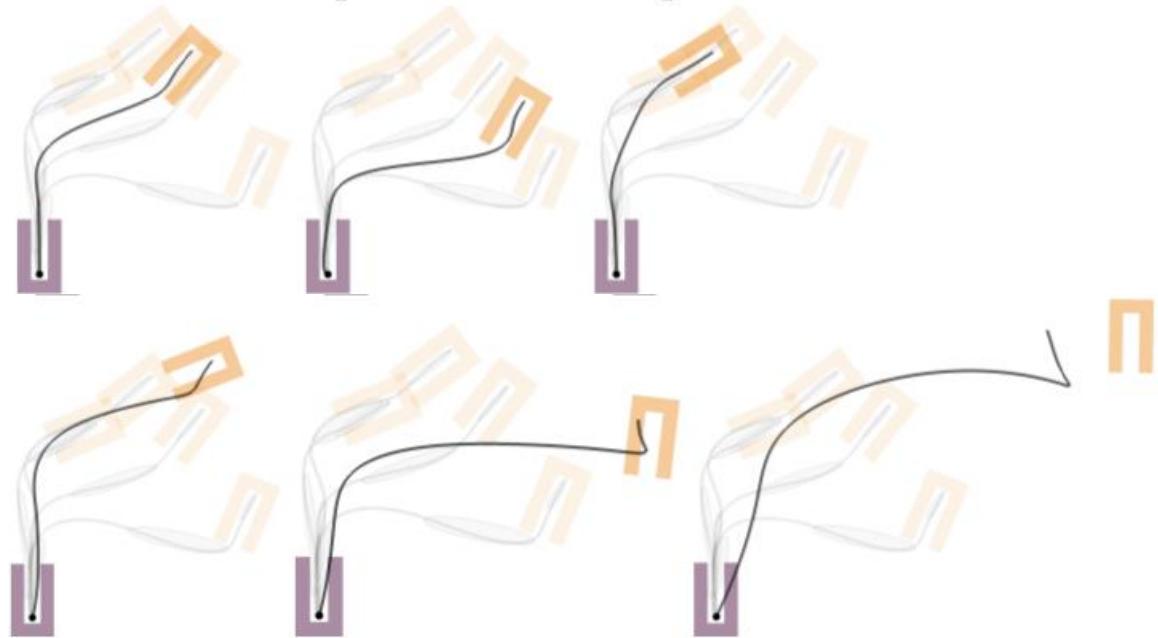
Regression with a context variable c :

- Learning of $\mathcal{P}(c, x)$
- Retrieval with $\mathcal{P}(x|c)$

Demonstrations



Reproduction attempts



→ Generic approach, but
limited generalization capability

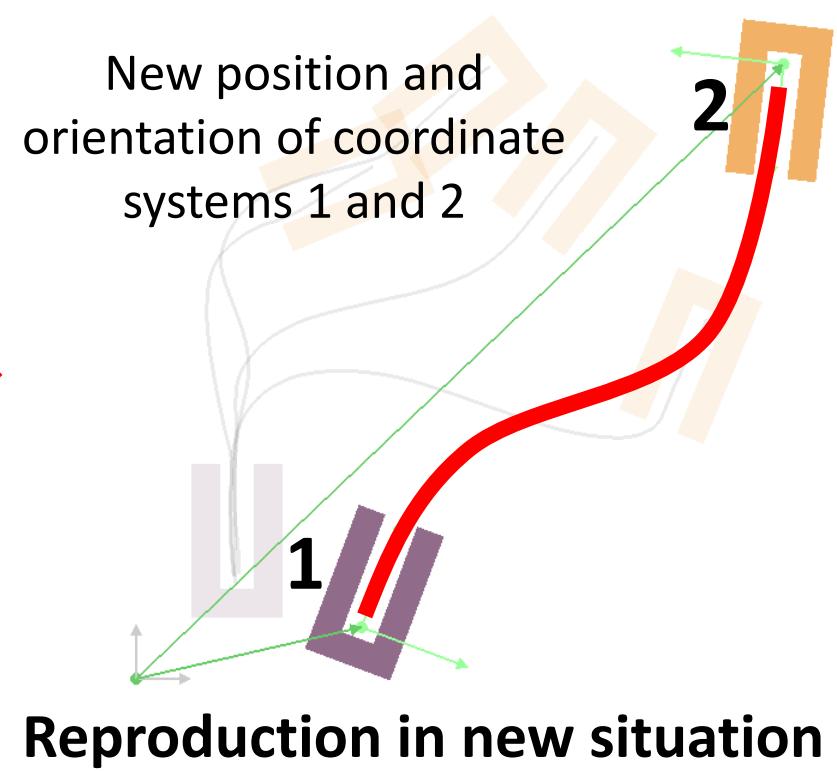
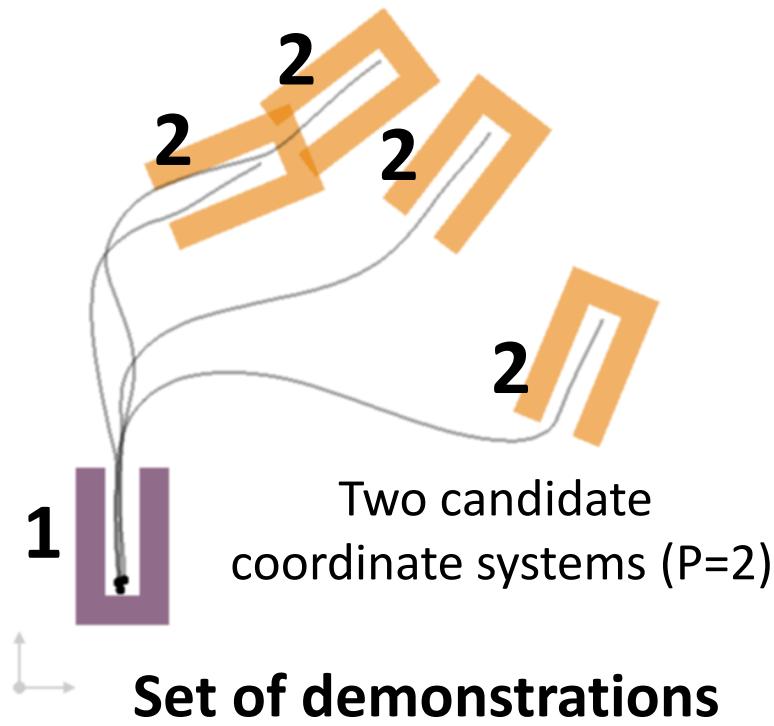
MPC/LQT in multiple coordinate systems

$$\min_u \sum_{t=1}^T \sum_{j=1}^P \left\| \boldsymbol{\mu}_t^{(j)} - \mathbf{x}_t \right\|_{Q_t^{(j)}}^2 + \left\| \mathbf{u}_t \right\|_{R_t}^2$$

Track path in coordinate system j

s.t. $\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t$ System dynamics

Use low control commands



MPC/LQT in multiple coordinate systems

$$\min_u \sum_{t=1}^T \sum_{j=1}^P \left\| \boldsymbol{\mu}_t^{(j)} - \mathbf{x}_t \right\|_{Q_t^{(j)}}^2 + \left\| \mathbf{u}_t \right\|_{R_t}^2$$

Track path in coordinate system j

Use low control commands

$$\text{s.t. } \mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t$$

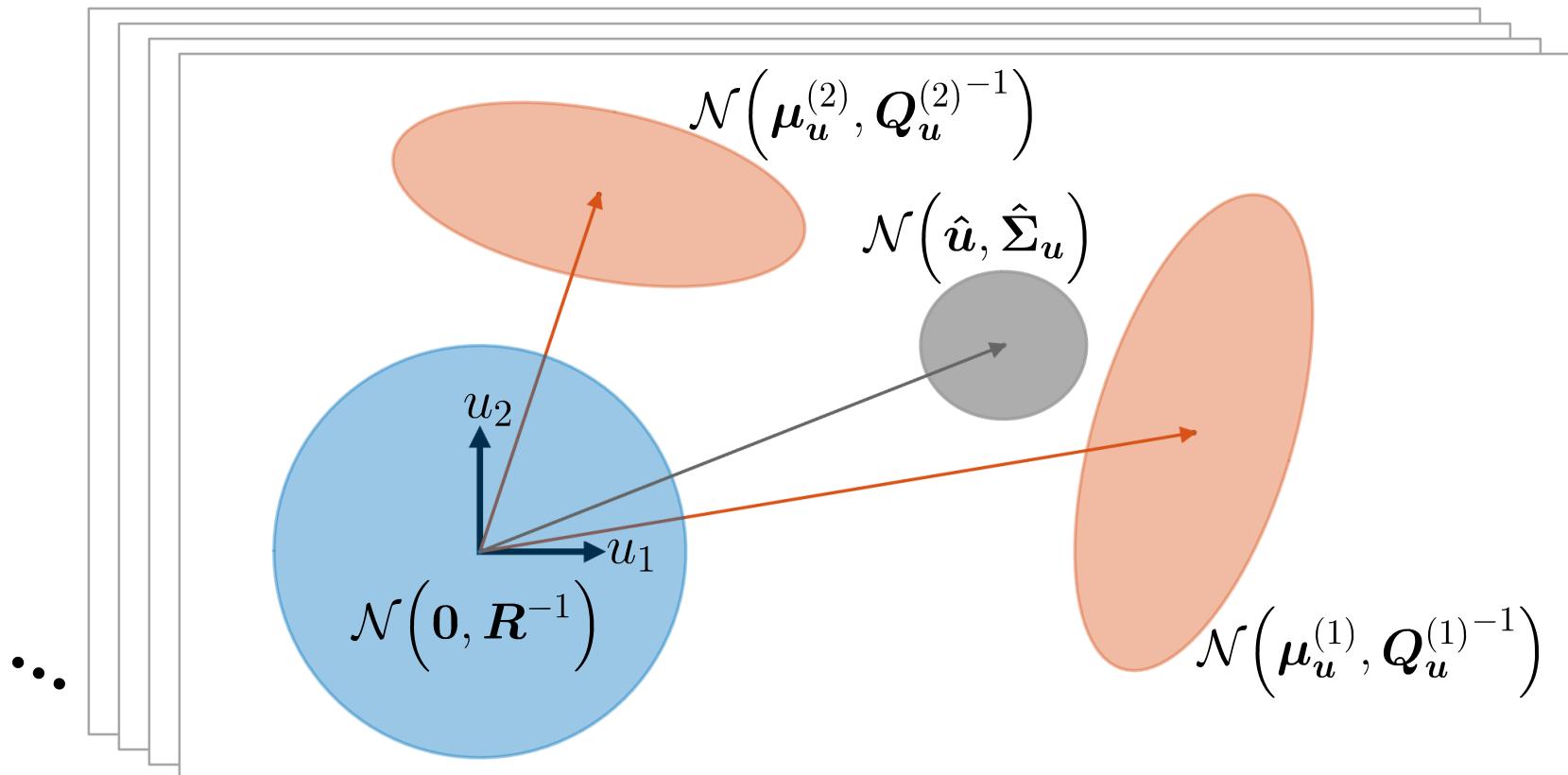
System dynamics



MPC/LQT in multiple coordinate systems

$$\min_u \sum_{t=1}^T \sum_{j=1}^P \left\| \boldsymbol{\mu}_t^{(j)} - \boldsymbol{x}_t \right\|_{Q_t^{(j)}}^2 + \left\| \boldsymbol{u}_t \right\|_{R_t}^2$$

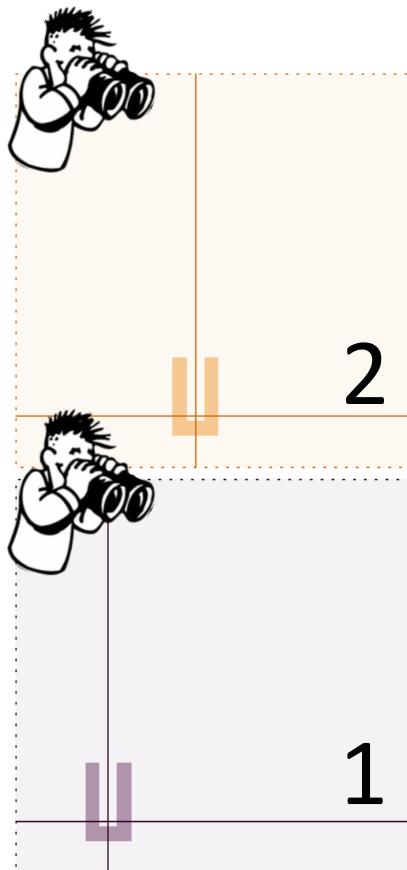
Track path in coordinate system j
Use low control commands



MPC/LQT in multiple coordinate systems

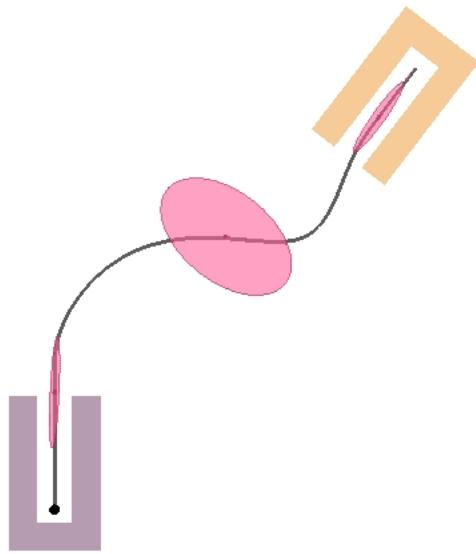
$$\min_u \sum_{t=1}^T \sum_{j=1}^P \left\| \boldsymbol{\mu}_t^{(j)} - \boldsymbol{x}_t \right\|_{Q_t^{(j)}}^2 + \left\| \boldsymbol{u}_t \right\|_{R_t}^2$$

In many robotics problems, the parameters describing the task or situation can be interpreted as coordinate systems

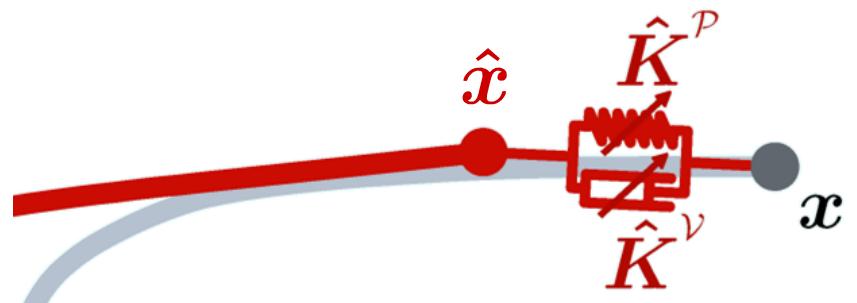


MPC/LQT in multiple coordinate systems

$$\min_u \sum_{t=1}^T \sum_{j=1}^P \left\| \boldsymbol{\mu}_t^{(j)} - \boldsymbol{x}_t \right\|_{Q_t^{(j)}}^2 + \left\| \boldsymbol{u}_t \right\|_{R_t}^2$$

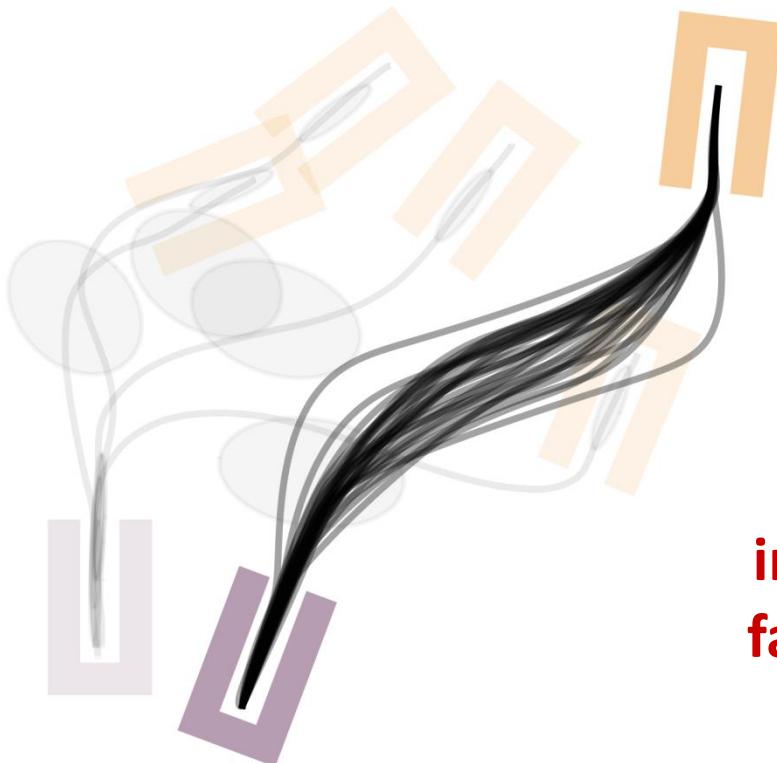


➔ **Learning of a controller**
(instead of learning a trajectory)
that adapts to new situations
while regulating the gains
according to the precision and
coordination required by the task



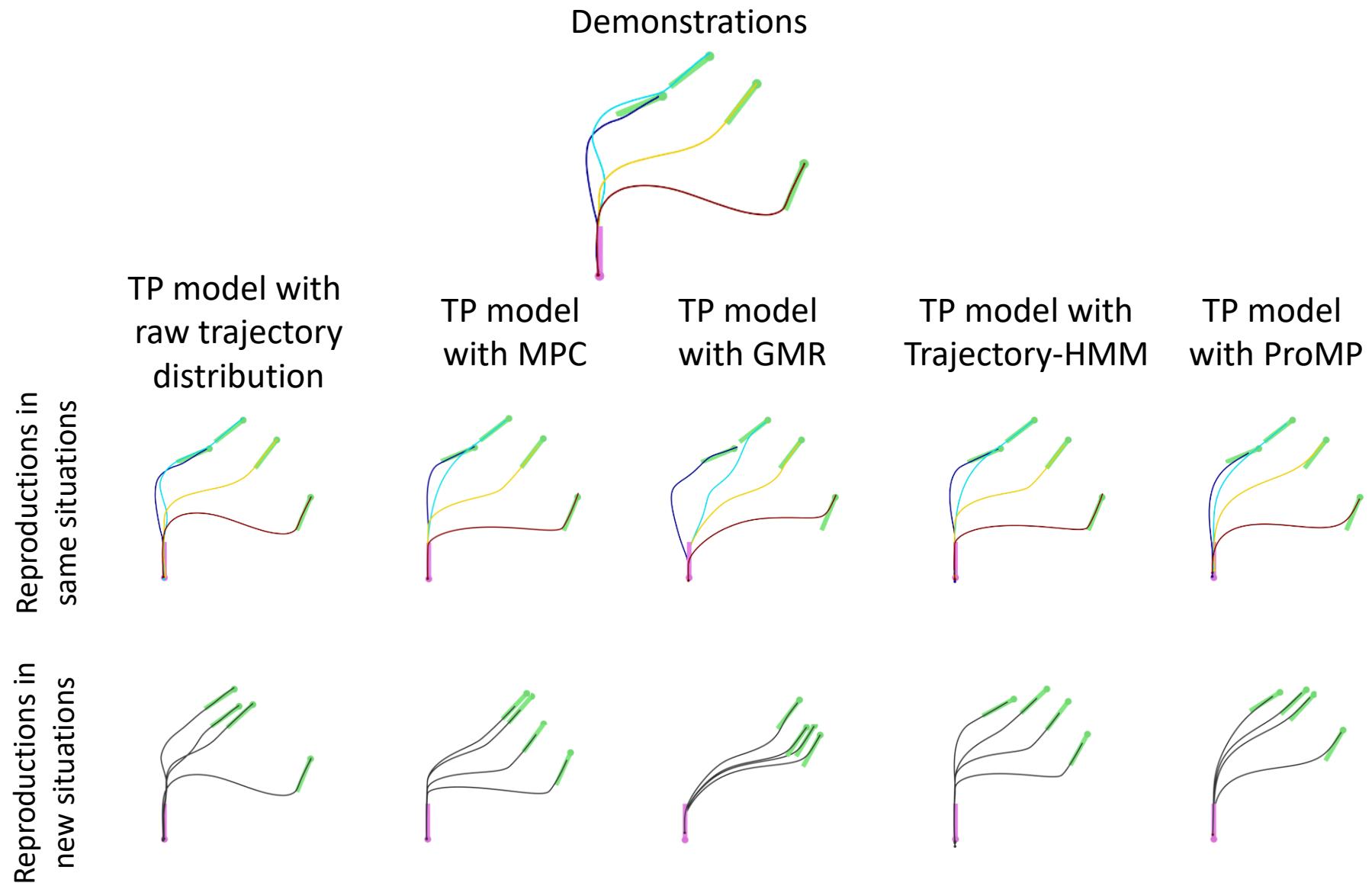
MPC/LQT in multiple coordinate systems

$$\min_u \sum_{t=1}^T \sum_{j=1}^P \left\| \boldsymbol{\mu}_t^{(j)} - \boldsymbol{x}_t \right\|_{Q_t^{(j)}}^2 + \left\| \boldsymbol{u}_t \right\|_{R_t}^2$$



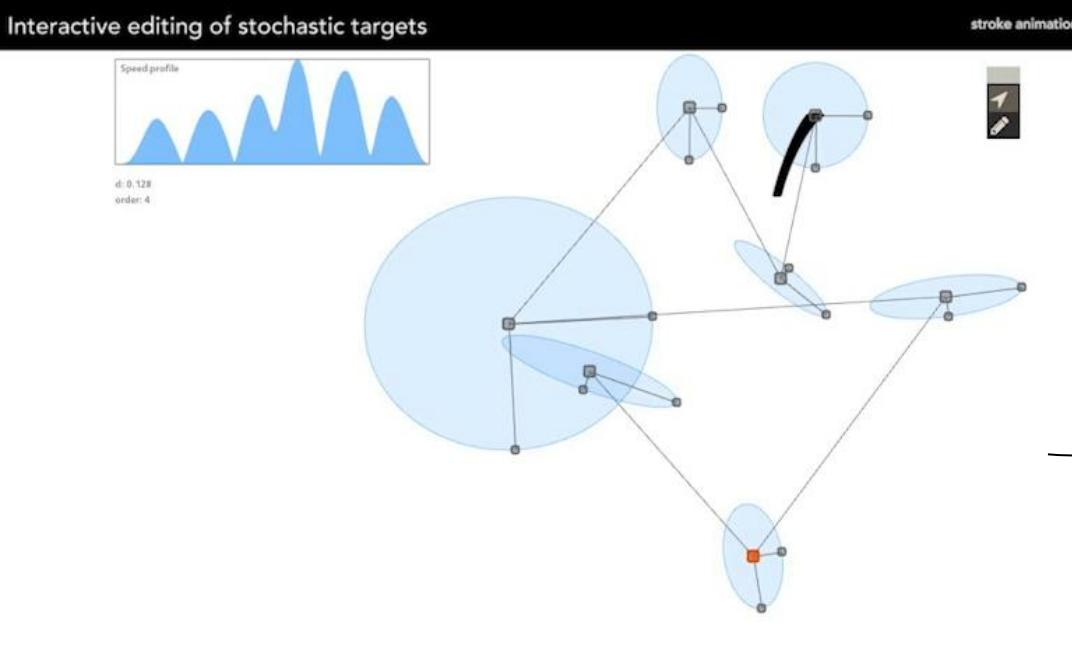
→ Retrieval of control commands
in the form of trajectory distributions,
facilitating exploration and adaptation
(in either control or state space)

Exploitation in other probabilistic models



Application examples

Application: Editing movements with variations



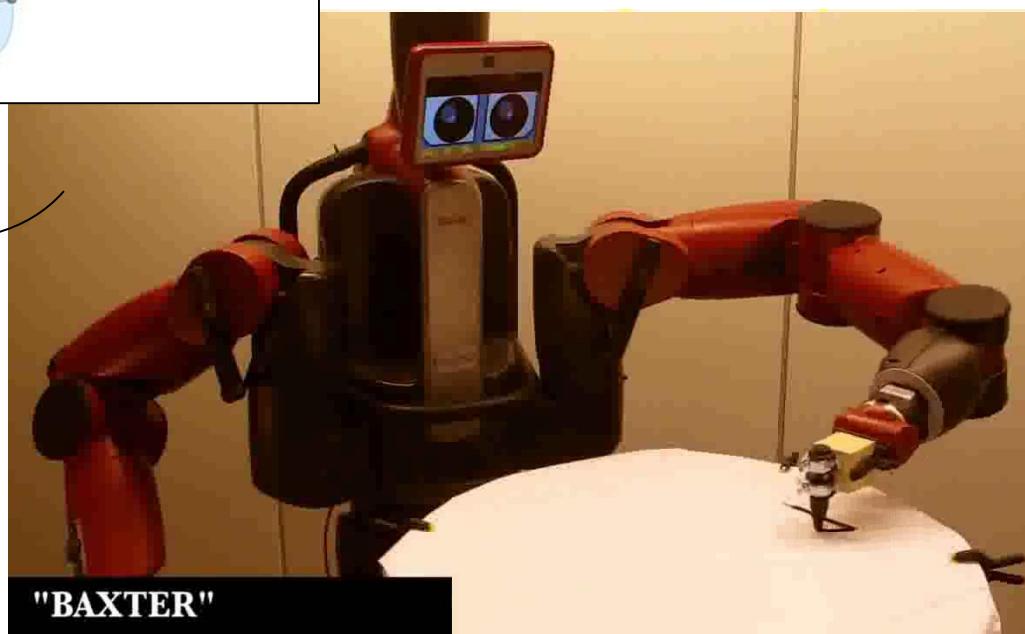
User interface to edit and generate natural and dynamic motions by considering variation and coordination

Compliant controller to retrieve safe and human-like motions



Daniel Berio

Frederic Fol Leymarie



[Berio, Calinon and Leymarie, IROS'2016] [Berio, Calinon and Leymarie , MOCO'2017]

Coordination and co-manipulation



[Silvérios et al., IROS'2015]



[Rozo et al., IROS'2015]



[Rozo et al., IEEE T-RO 32(3), 2016]



ISTITUTO
ITALIANO DI
TECNOLOGIA

Assistive dressing



SNSF, CHIST-ERA (2015-2018)
<https://i-dress-project.eu>



Shared control



DexROV

spaceapplications
SERVICES

GRAALtech

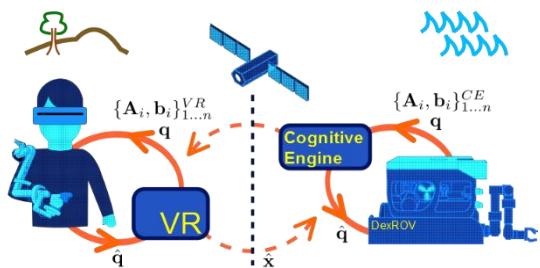
JACOBS
UNIVERSITY

comex

ISME
Integrated Systems for Marine Environment

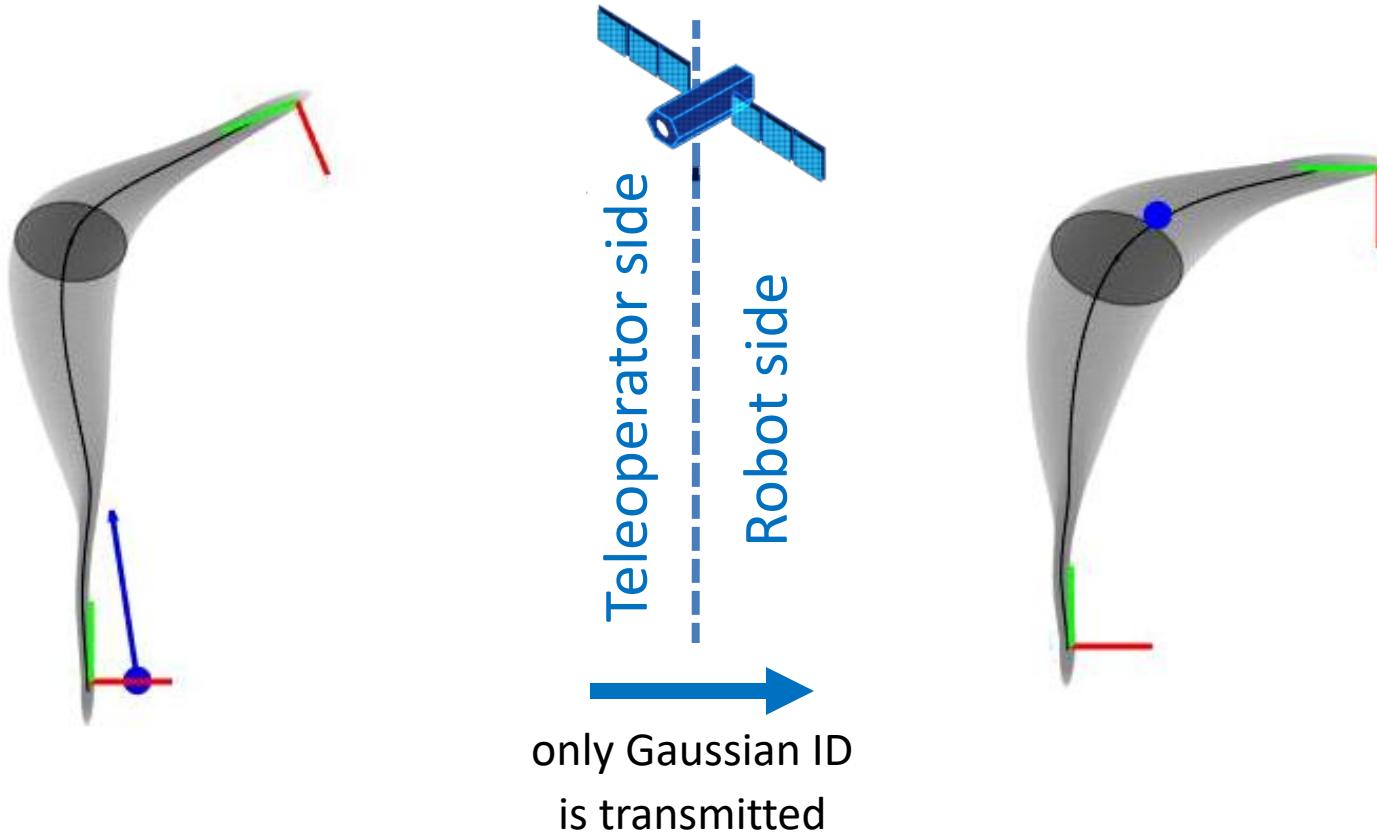
EJR-QUARTZ

idiap
RESEARCH INSTITUTE



<http://dexrov.eu>
EC, H2020 (2015-2018)

Shared control



Adaptation to different object shapes

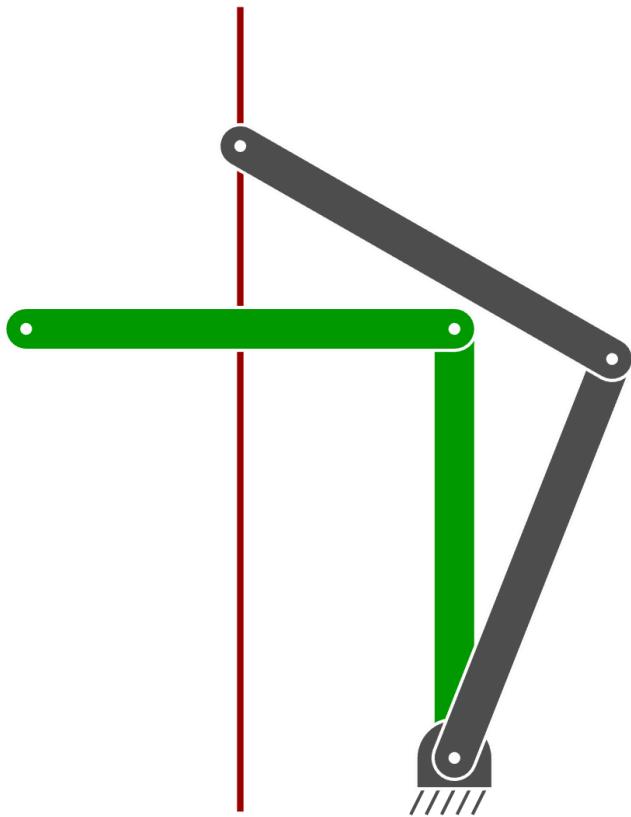


Coordinate
system
as task
parameter



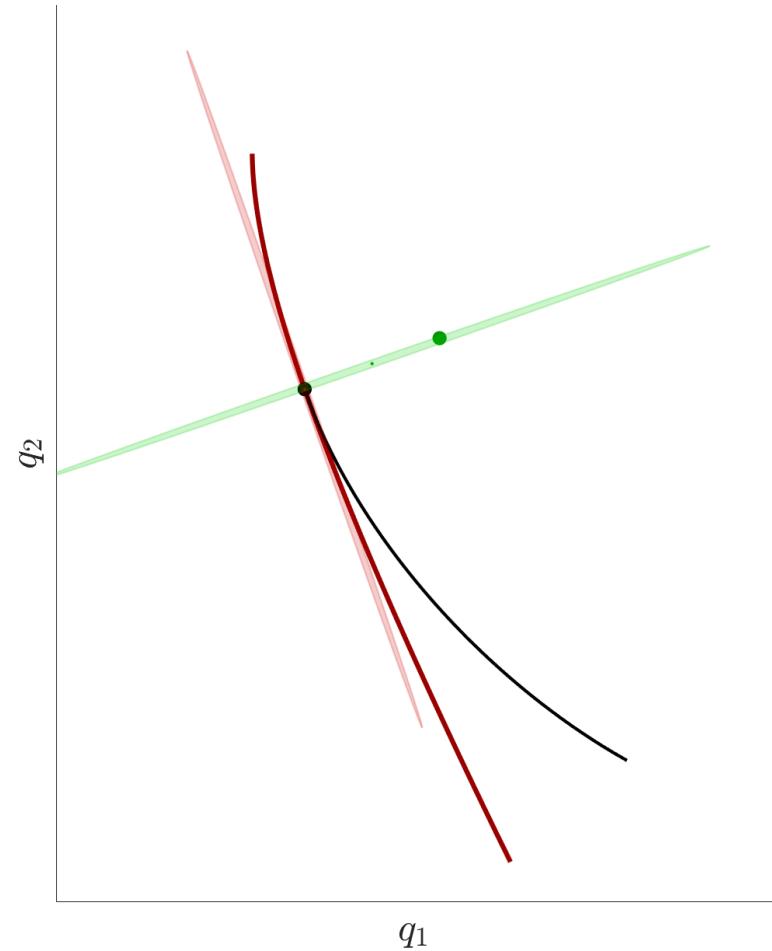
Learning tasks prioritization

$$\begin{aligned}\hat{\mathbf{u}} &= \arg \min_{\mathbf{u}} \|\mathbf{u} - \mathbf{J}^\dagger \dot{\mathbf{x}}\|_{\mathbf{J}^\dagger \mathbf{J}}^2 + \|\mathbf{u} - \dot{\mathbf{q}}\|_{\mathbf{I} - \mathbf{J}^\dagger \mathbf{J}}^2 \\ &= \mathbf{J}^\dagger \dot{\mathbf{x}} + (\mathbf{I} - \mathbf{J}^\dagger \mathbf{J}) \dot{\mathbf{q}}\end{aligned}$$



Principal task:
track horizontal reference

Secondary task:
track desired posture

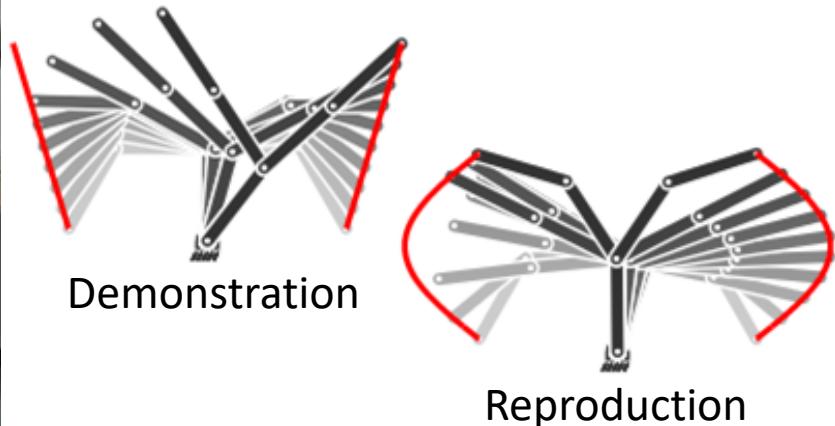


Learning tasks prioritization

Demonstration



Priority on left hand



$$\hat{\dot{q}} = \underbrace{\begin{bmatrix} J_1^\dagger & N_1 J_2^\dagger \end{bmatrix}}_{\text{Candidate hierarchy } A_1} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix}$$

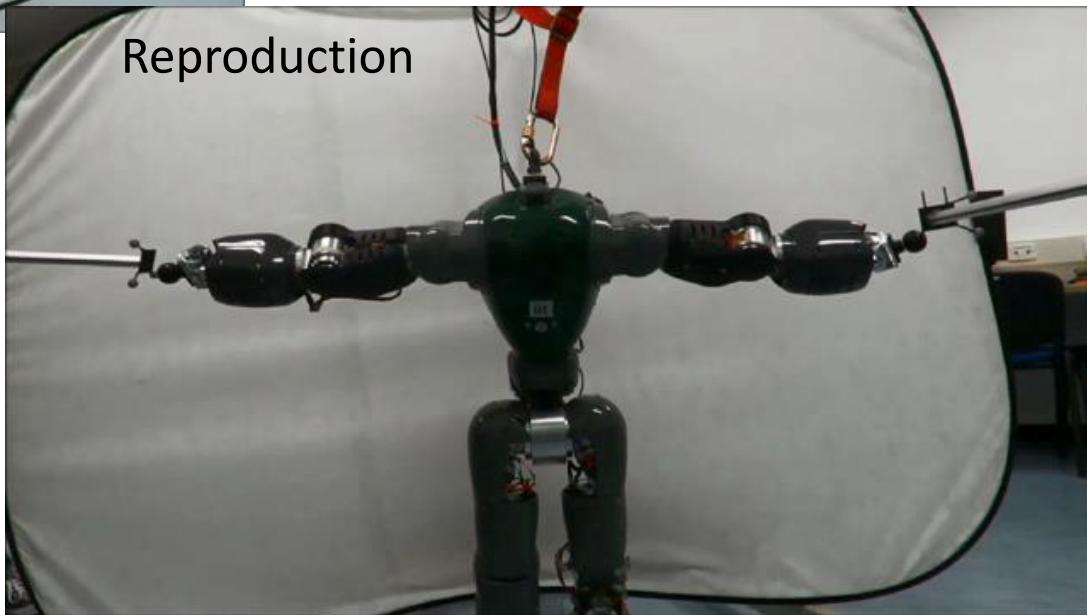
Candidate hierarchy A_1

$$\hat{\dot{q}} = \underbrace{\begin{bmatrix} N_2 J_1^\dagger & J_2^\dagger \end{bmatrix}}_{\text{Candidate hierarchy } A_2} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix}$$

Candidate hierarchy A_2

[Silvério, Calinon, Rozo and Caldwell, IEEE T-RO 2019] [Calinon, ISRR'15]

Reproduction



Learning tasks prioritization

Demonstration



Priority on right hand



Demonstration



Reproduction

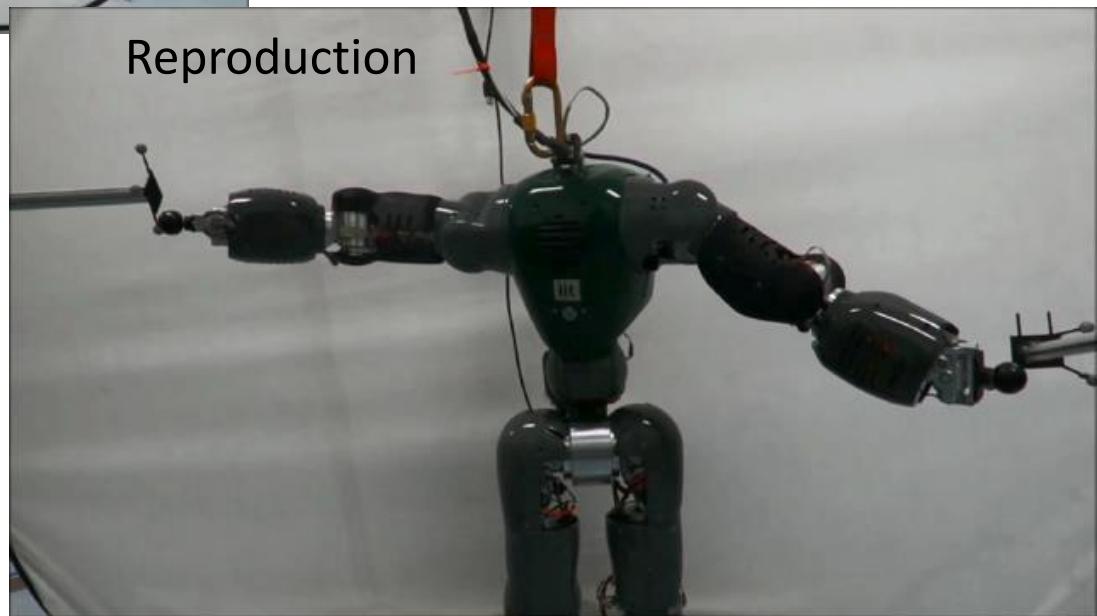
$$\hat{\dot{q}} = \underbrace{\begin{bmatrix} J_1^\dagger & N_1 J_2^\dagger \end{bmatrix}}_{\text{Candidate hierarchy } A_1} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix}$$

Candidate hierarchy A_1

$$\hat{\dot{q}} = \underbrace{\begin{bmatrix} N_2 J_1^\dagger & J_2^\dagger \end{bmatrix}}_{\text{Candidate hierarchy } A_2} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix}$$

Candidate hierarchy A_2

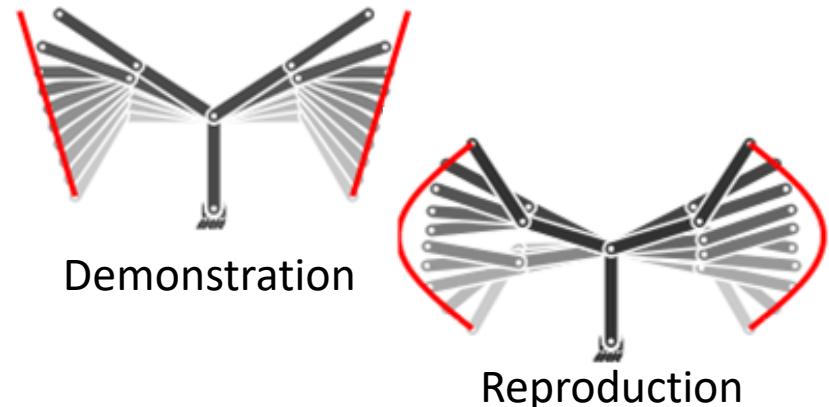
Reproduction



Learning tasks prioritization



Equal priority

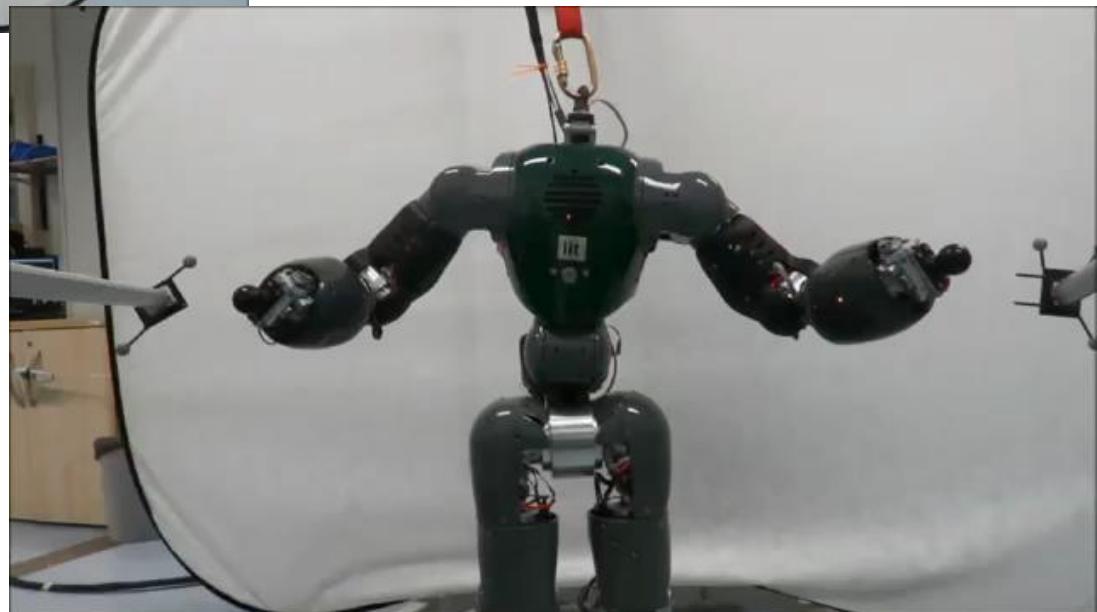


$$\hat{\dot{q}} = \underbrace{\begin{bmatrix} J_1^\dagger & N_1 J_2^\dagger \end{bmatrix}}_{\text{Candidate hierarchy } A_1} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix}$$

Candidate hierarchy A_1

$$\hat{\dot{q}} = \underbrace{\begin{bmatrix} N_2 J_1^\dagger & J_2^\dagger \end{bmatrix}}_{\text{Candidate hierarchy } A_2} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix}$$

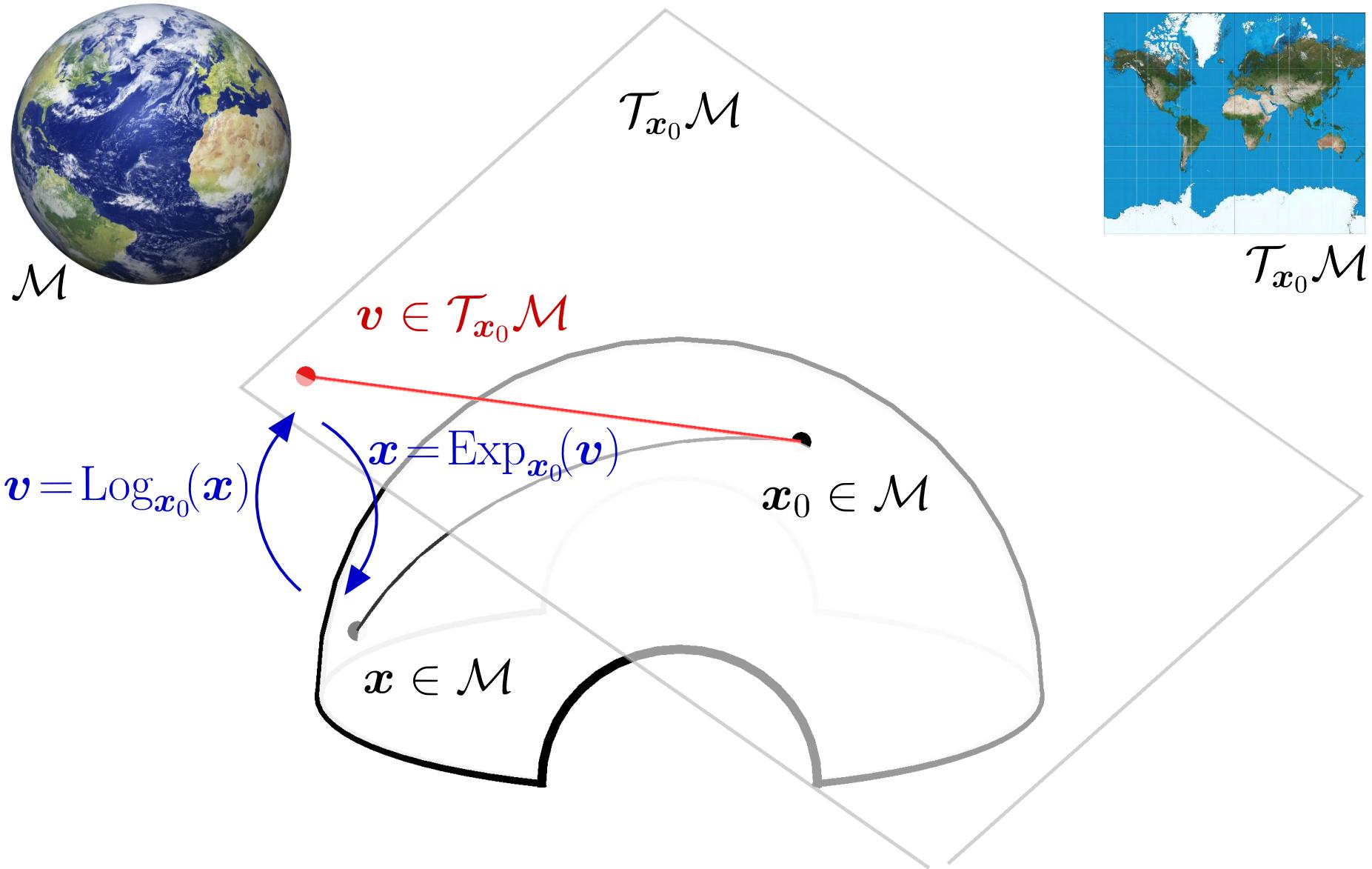
Candidate hierarchy A_2



Outline

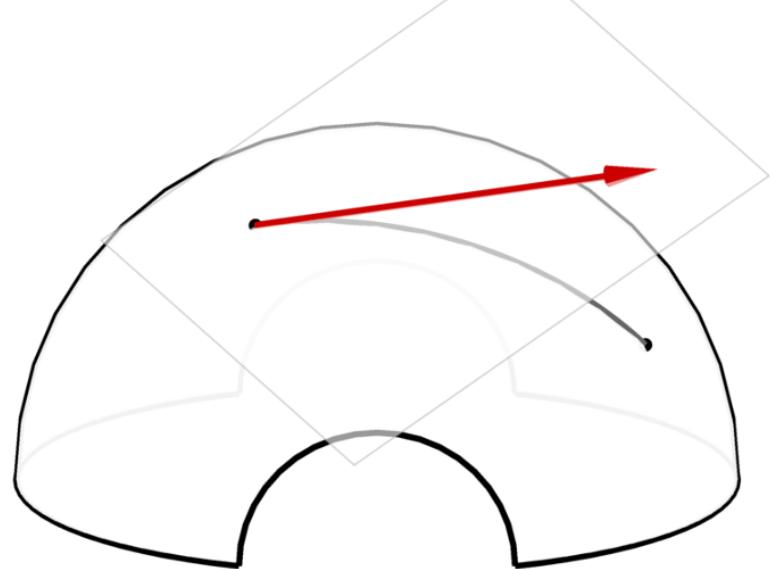
- **Superposition with basis functions**
 - Bezier curves
 - Locally weighted regression (LWR)
 - Gaussian mixture regression (GMR)
 - Fourier series for periodic motion and ergodic control
- **Dynamical movement primitives (DMP)**
 - Probabilistic movement primitives (ProMP)
- **Superposition Vs fusion**
 - Product of Gaussians
- **Model predictive control (MPC)**
 - Linear quadratic tracking (LQT)
 - Task-parameterized movement models
- **Differential geometry**
 - Riemannian manifolds

Riemannian geometry: use of tangent spaces

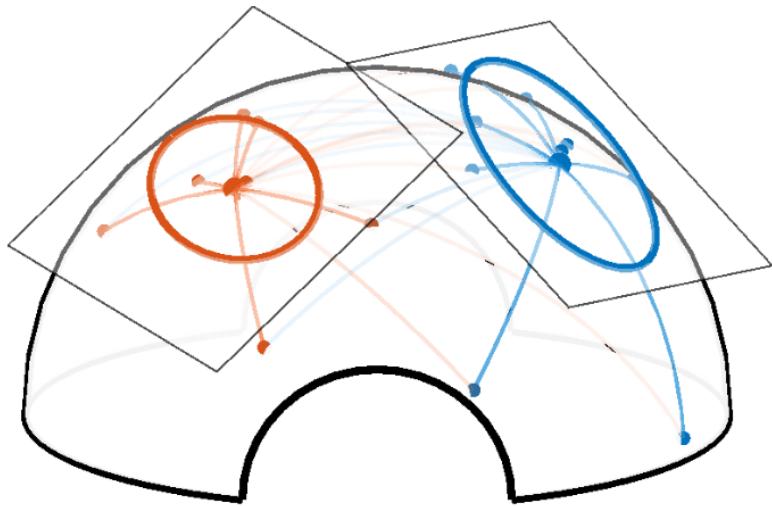


Sphere and orientation (unit quaternion) manifolds

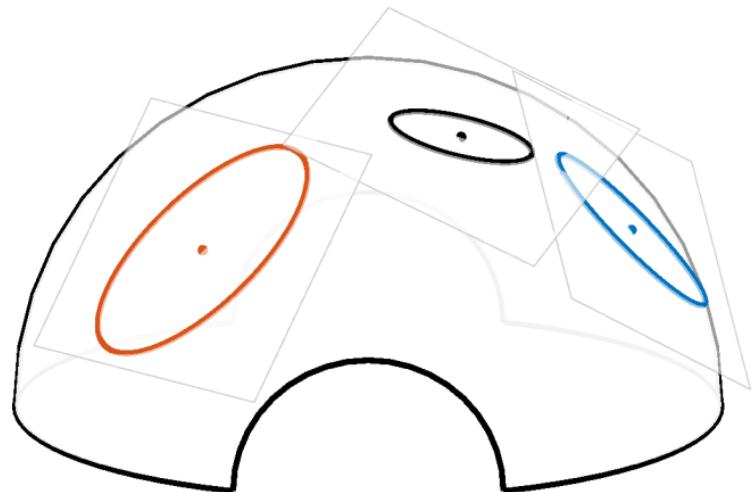
Riemannian manifolds



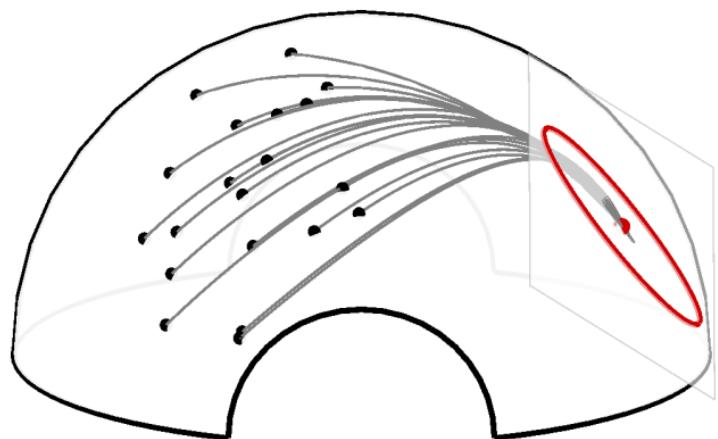
Interpolation and extrapolation



Clustering and distribution



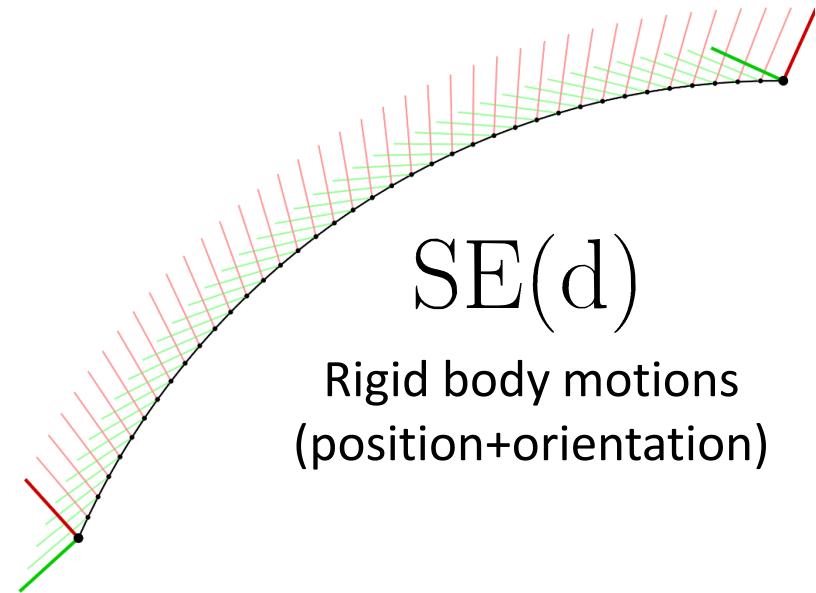
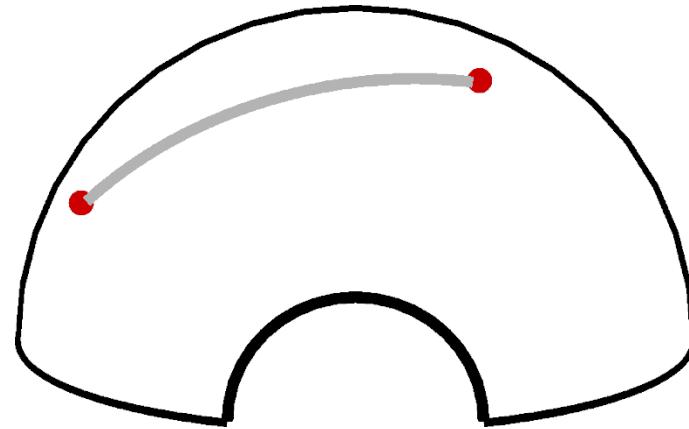
Fusion of sensing/control information



Linear quadratic tracking

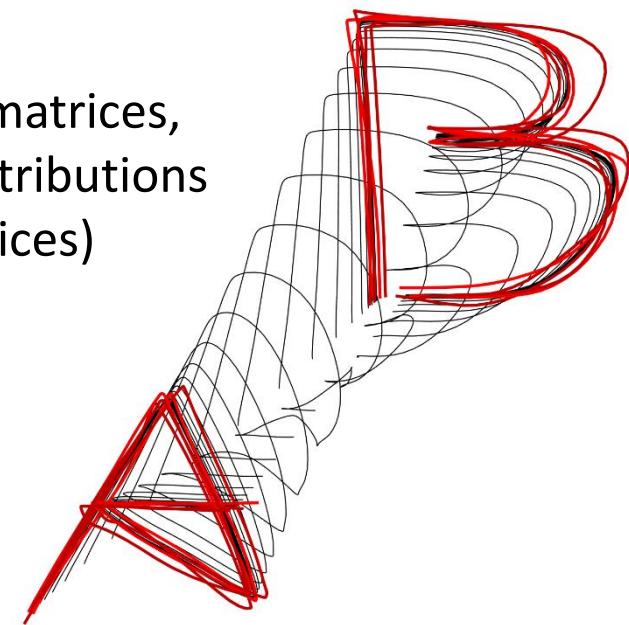
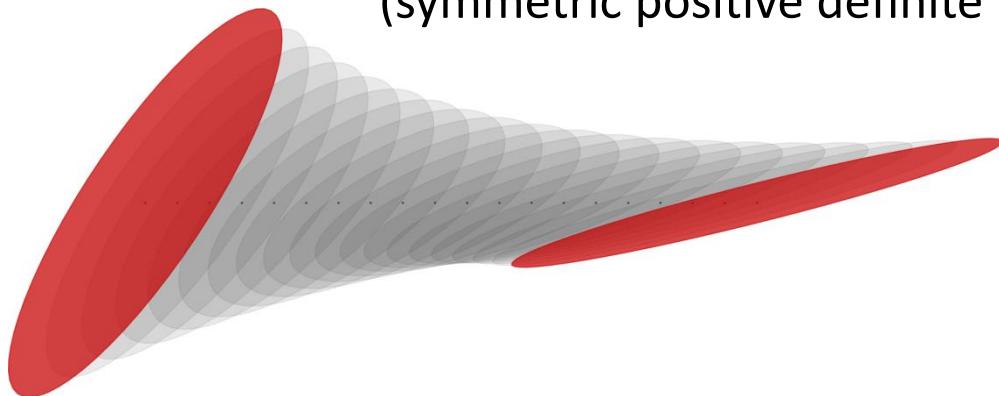
Interpolation on Riemannian manifolds

Orientation
(unit quaternions) S^d

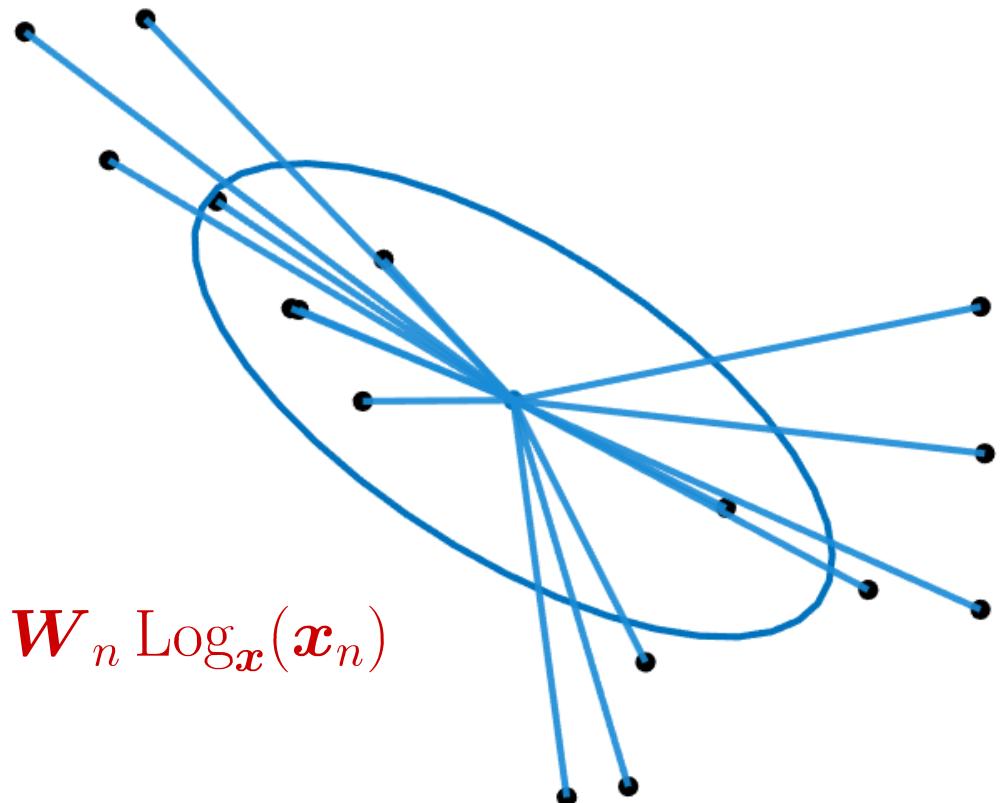


$SE(d)$
Rigid body motions
(position+orientation)

S^d_{++} Covariance features, inertia and gain matrices,
manipulability ellipsoids, trajectory distributions
(symmetric positive definite matrices)



Statistics on Riemannian manifolds



$$\boldsymbol{\mu} = \arg \min_{\boldsymbol{x}} \sum_{n=1}^N \frac{1}{2} \text{Log}_{\boldsymbol{x}}(\boldsymbol{x}_n)^\top \boldsymbol{W}_n \text{Log}_{\boldsymbol{x}}(\boldsymbol{x}_n)$$

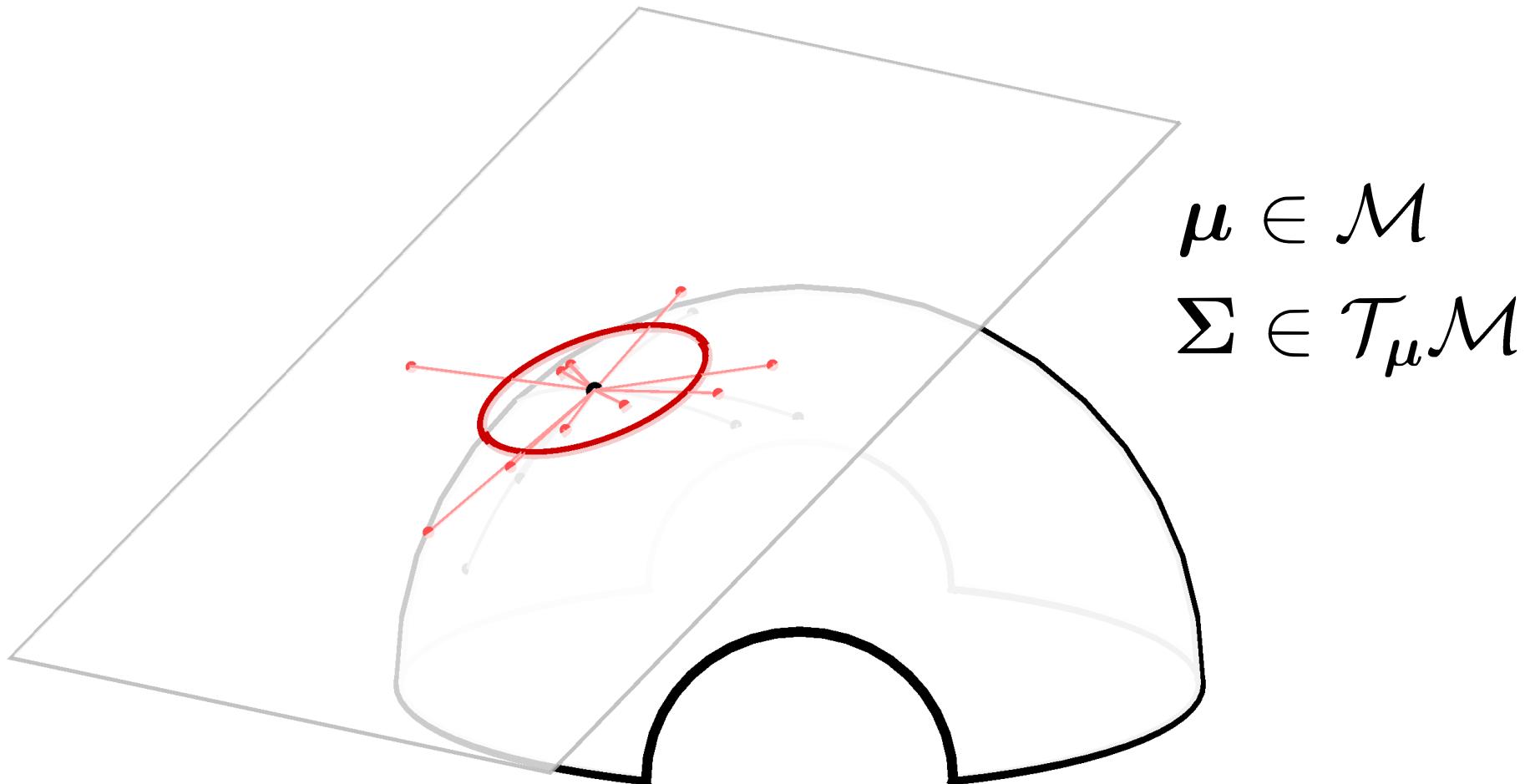
$$\boldsymbol{\mu} = \frac{1}{N} \sum_{n=1}^N \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma} = \frac{1}{N} \sum_{n=1}^N (\boldsymbol{x}_n - \boldsymbol{\mu})(\boldsymbol{x}_n - \boldsymbol{\mu})^\top$$

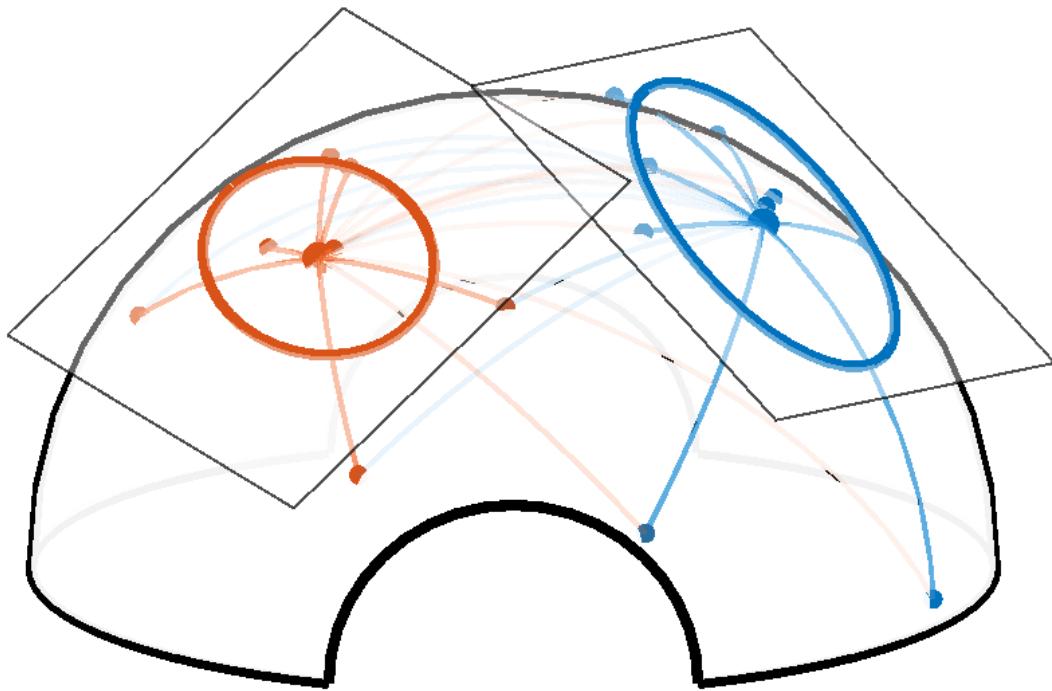
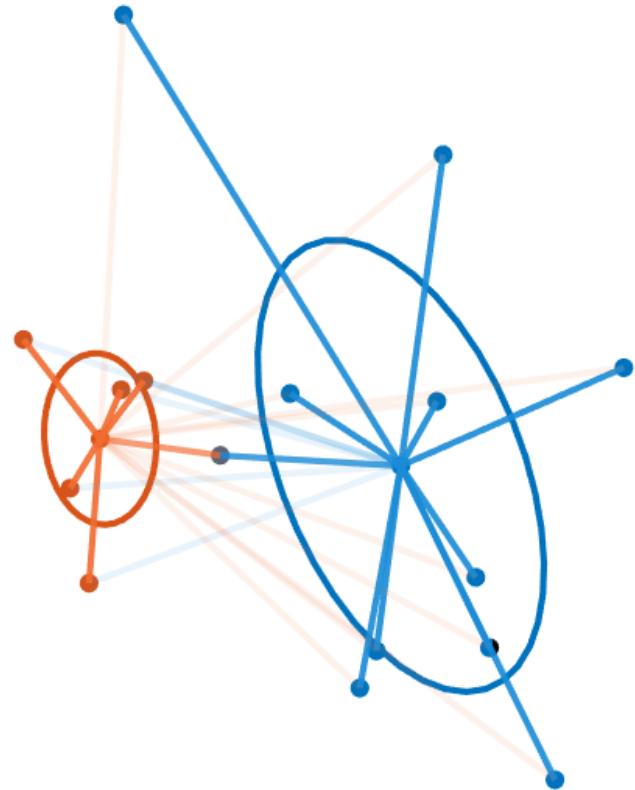
N datapoints

Riemannian geometry: Gaussian distribution

$$\mathcal{N}_{\mathcal{M}}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}\text{Log}_{\boldsymbol{\mu}}(\mathbf{x})^\top \boldsymbol{\Sigma}^{-1} \text{Log}_{\boldsymbol{\mu}}(\mathbf{x})\right)$$



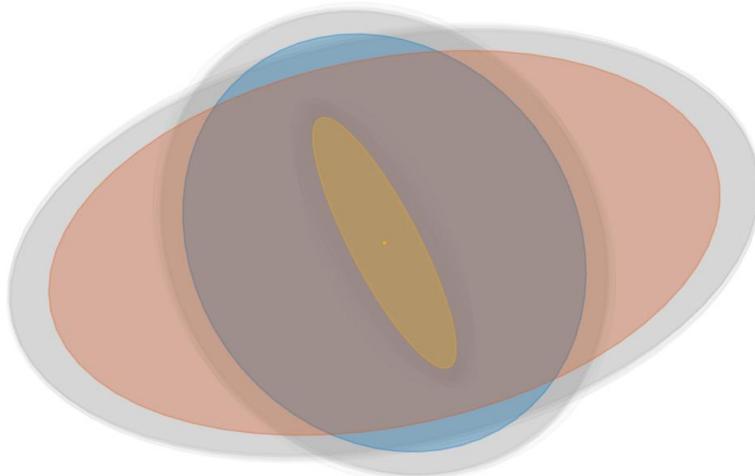
Clustering on Riemannian manifolds



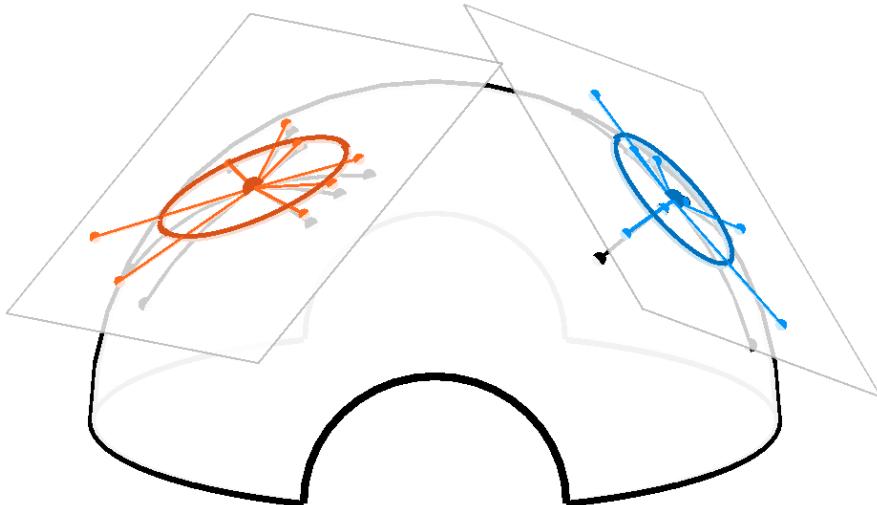
$$\mathcal{P}(\mathbf{x}_n) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

K Gaussians
N datapoints

Clustering on Riemannian manifolds

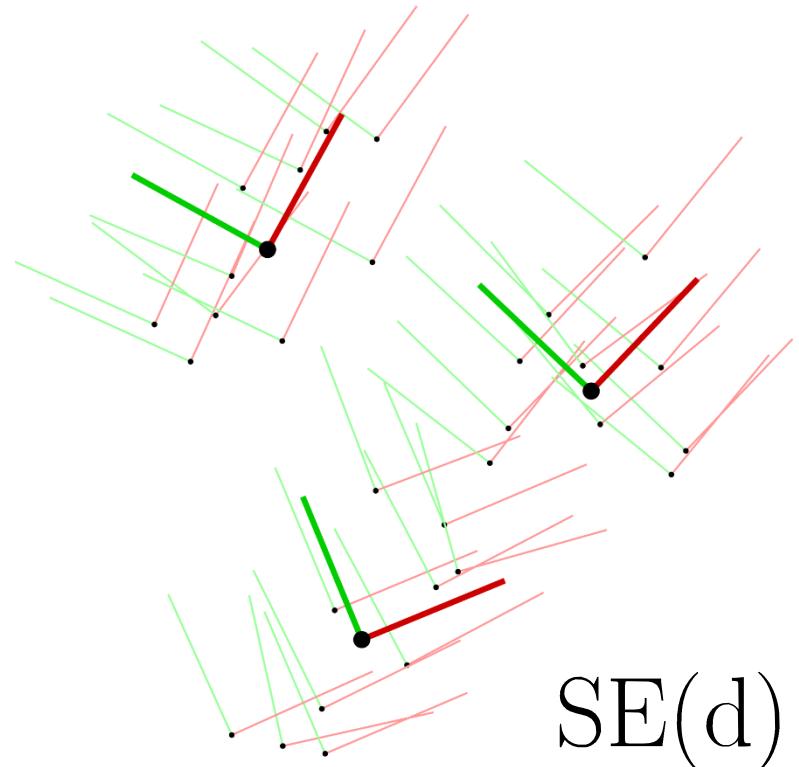


Orientation
(unit quaternions) \mathcal{S}^d



\mathcal{S}_{++}^d

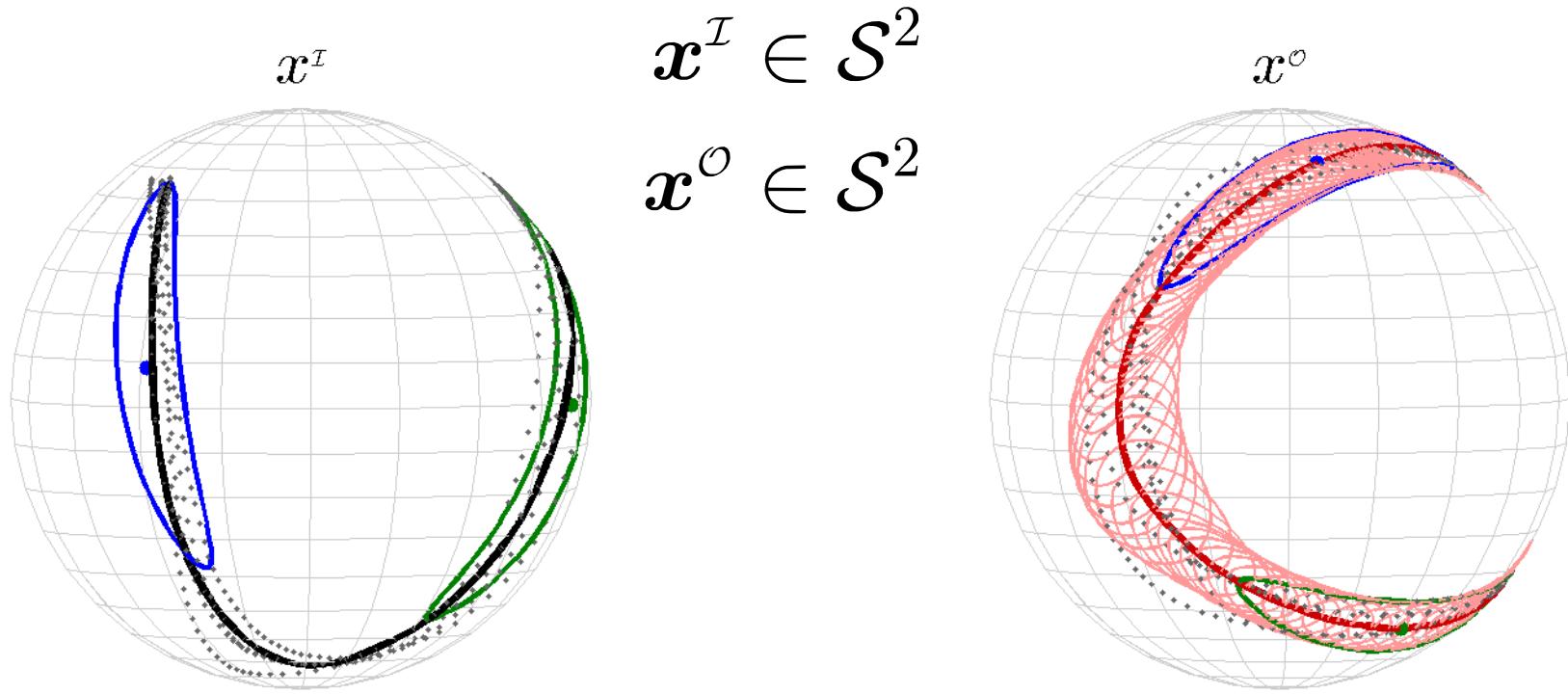
Covariance features, inertia and gain matrices, manipulability ellipsoids, trajectory distributions (symmetric positive definite matrices)



$SE(d)$
Rigid body motions
(position+orientation)

Regression on Riemannian manifolds

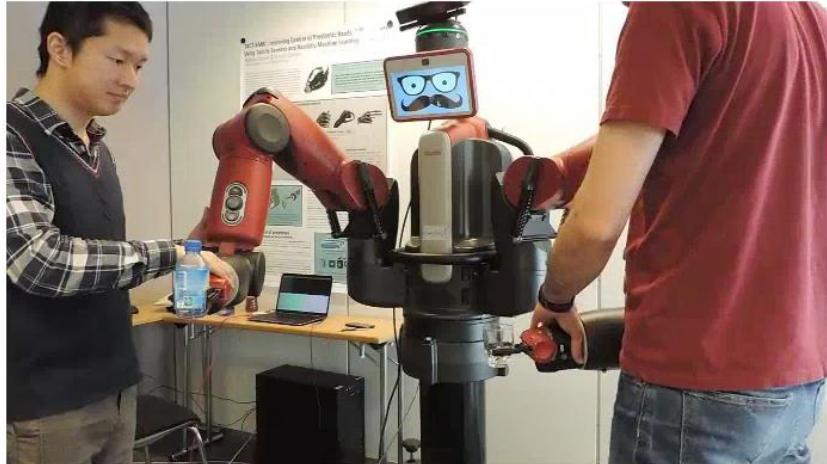
Gaussian mixture regression (GMR) to compute $\mathcal{P}(x^o|x^i)$
from the joint distribution $\mathcal{P}(x^i, x^o)$ encoded as a GMM



→ Regression for orientation data (unit quaternions on \mathcal{S}^3)

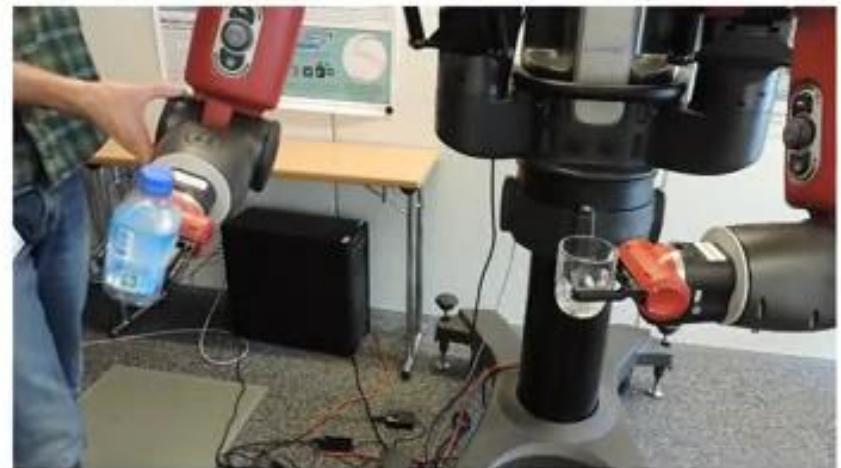
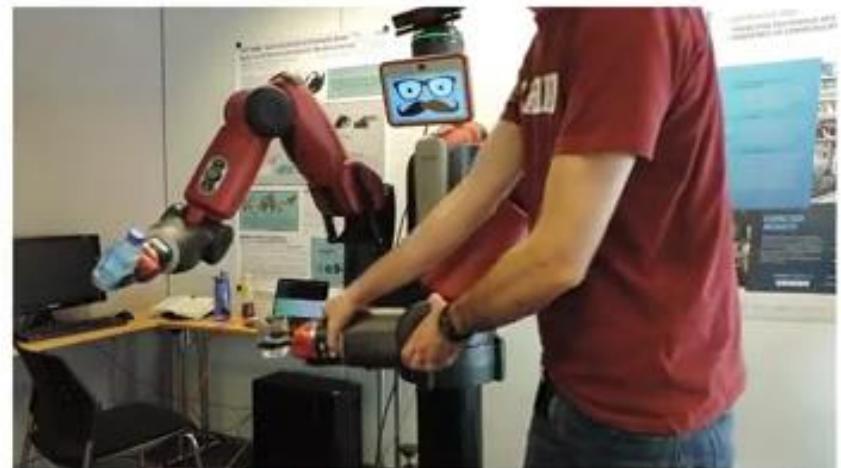
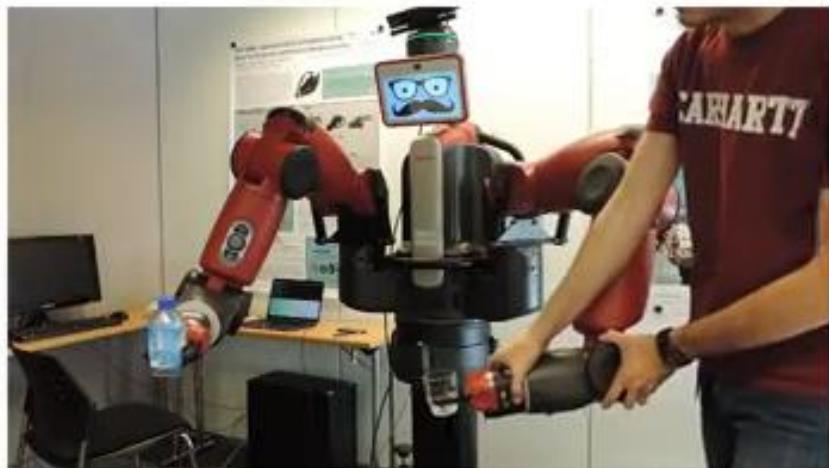
Regression with orientation and position data

Four demonstrations of coordinated bimanual movement



Regression with orientation and position data

Four reproductions with perturbations by the user



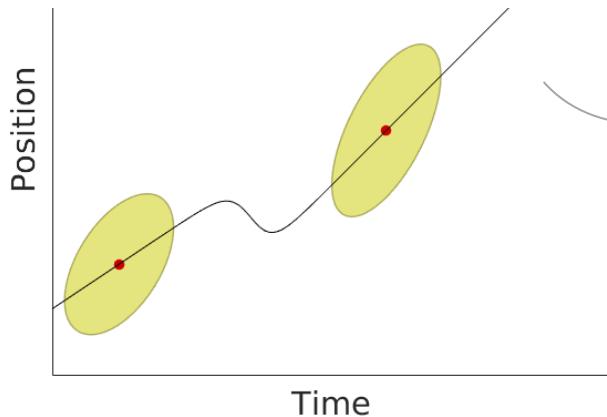
Regression with covariance features

Gaussian mixture regression (GMR) with SPD datapoints

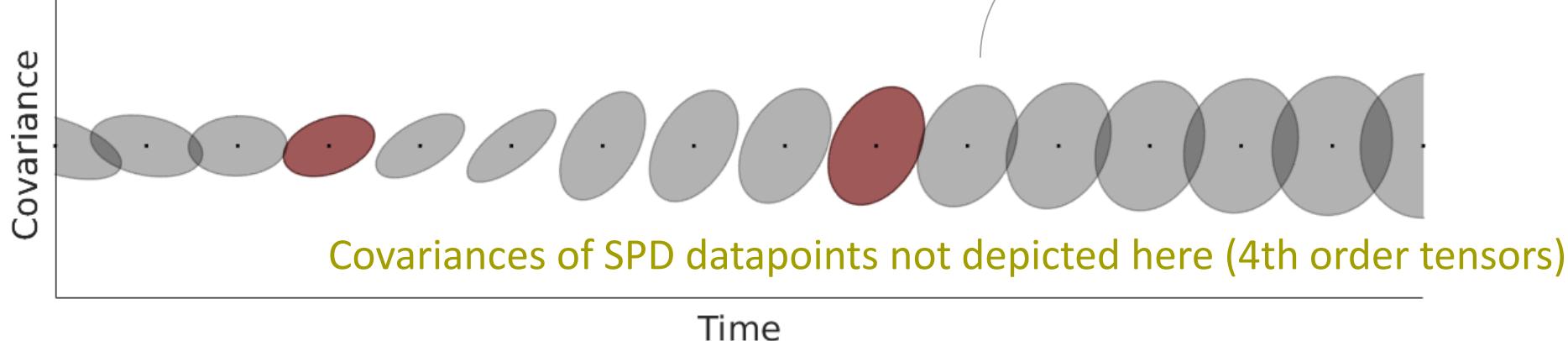
● centers

● covariances

$$\begin{aligned} \boldsymbol{x}^{\mathcal{I}} &\in \mathbb{R} \\ \boldsymbol{x}^o &\in \mathbb{R} \end{aligned}$$



$$\begin{aligned} \boldsymbol{x}^{\mathcal{I}} &\in \mathbb{R} \\ \boldsymbol{x}^o &\in \mathcal{S}_{++}^2 \end{aligned}$$

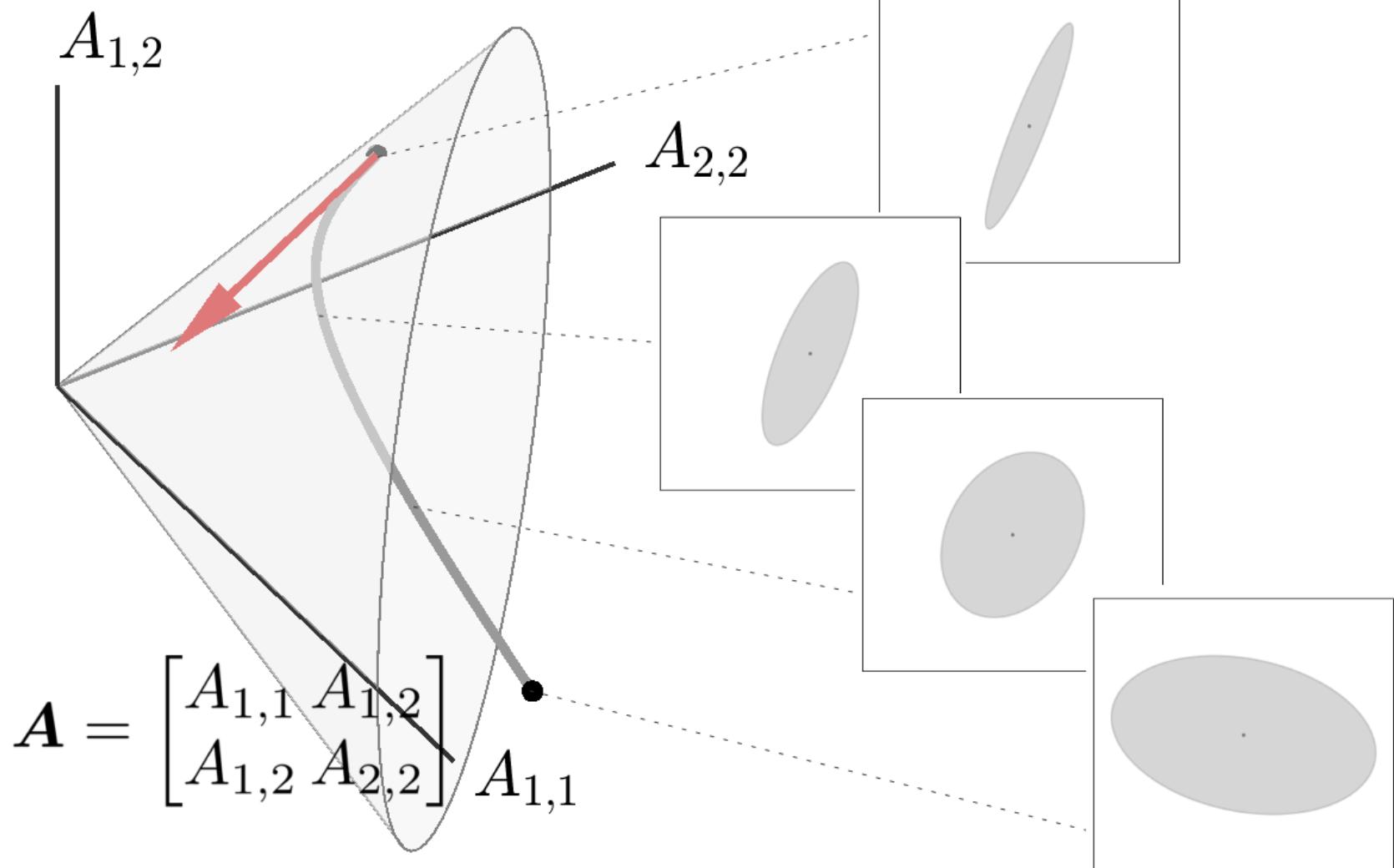


Covariances of SPD datapoints not depicted here (4th order tensors)

Time

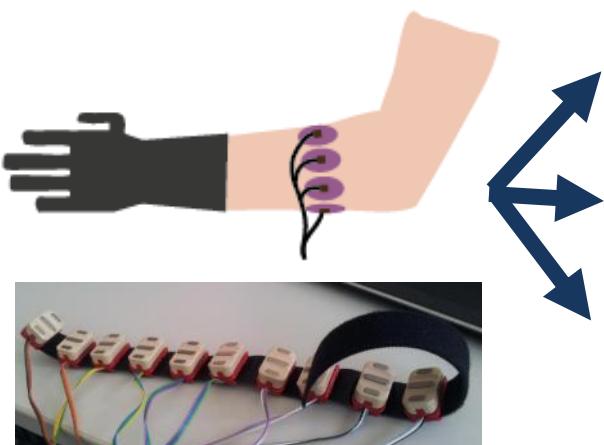
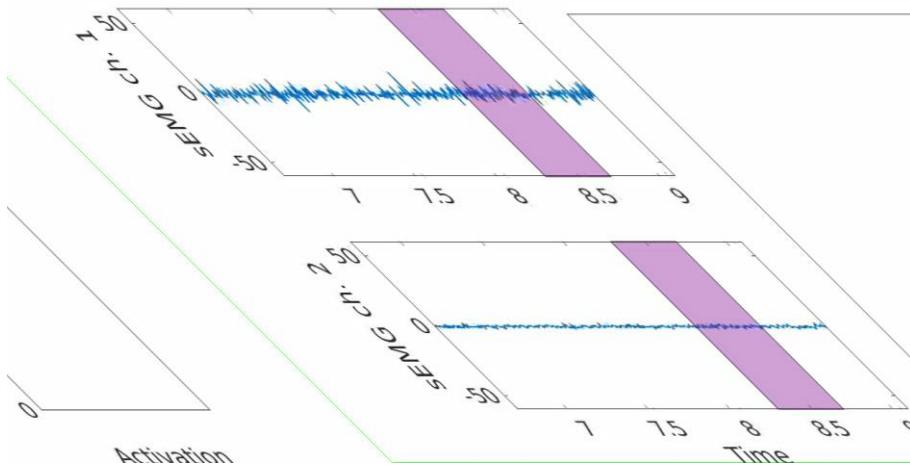
→ Predicting the temporal change of covariance features

Regression with covariance features



Symmetric positive definite (SPD) matrix manifold

Regression with electromyography data



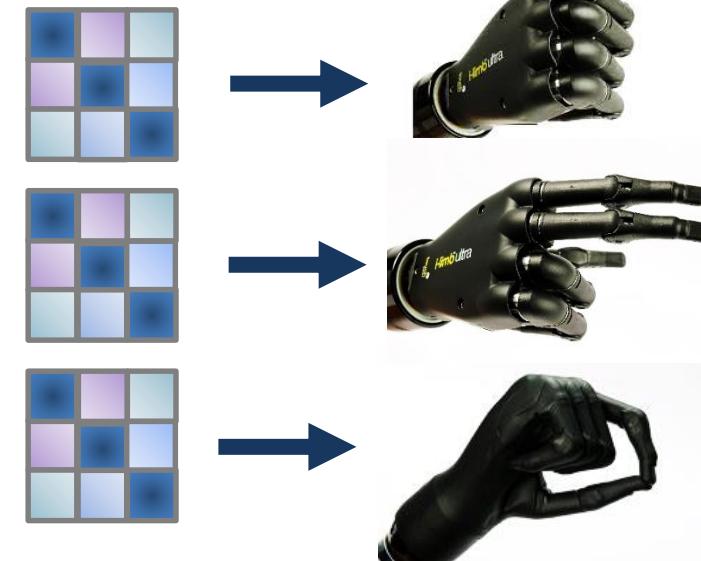
Surface electromyography
(sEMG) measurements

Transformation in **spatial
covariances** (SPD matrices)

Control of the
corresponding hand pose



SNSF, D-A-CH (2016-2019)



[Jaquier and Calinon, IROS 2017]

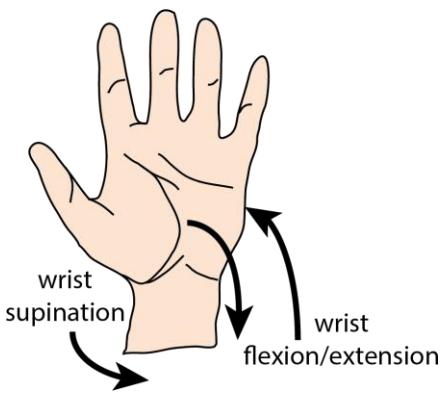
Regression with electromyography data



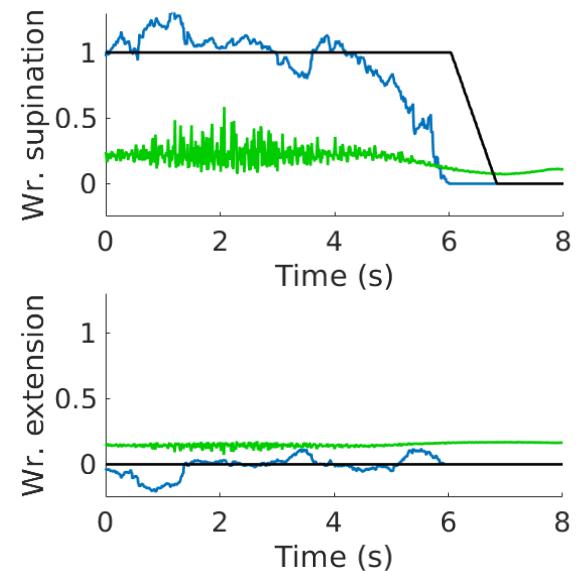
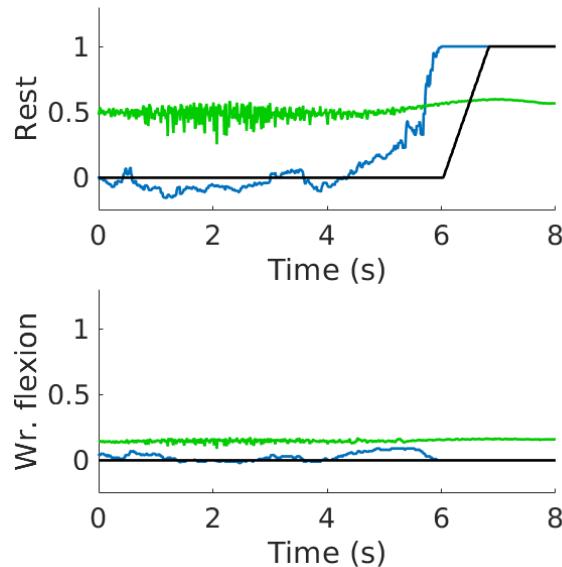
sEMG data from Ninapro database processed as spatial covariances:

$$\text{Input} \in \mathcal{S}_{++}^{12}$$

$$\text{Output} \in \mathbb{R}^4$$

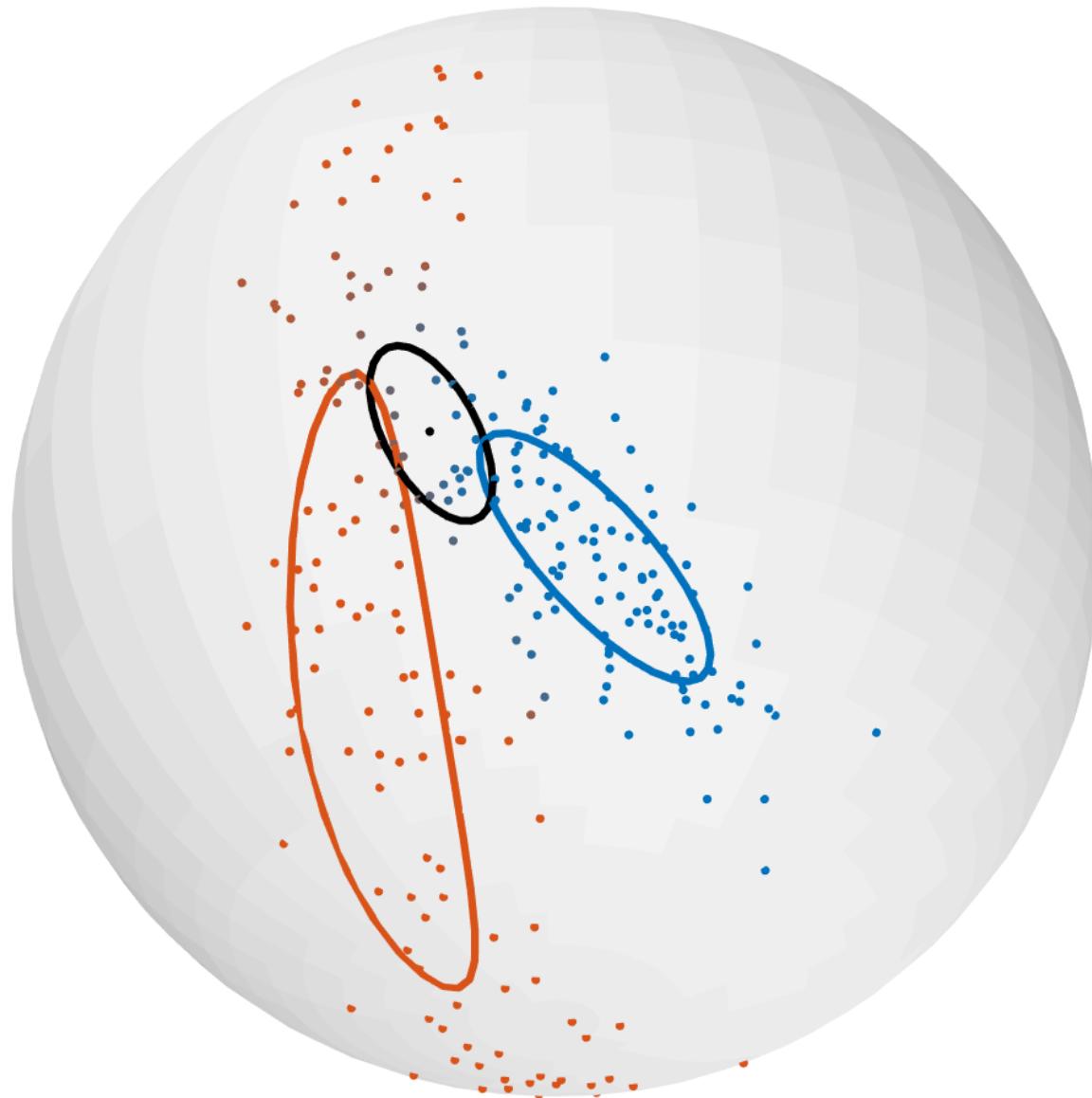


- Reference
- Standard regression
- Regression on SPD manifold



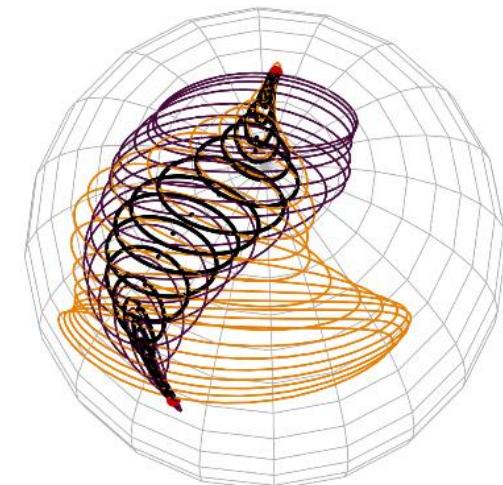
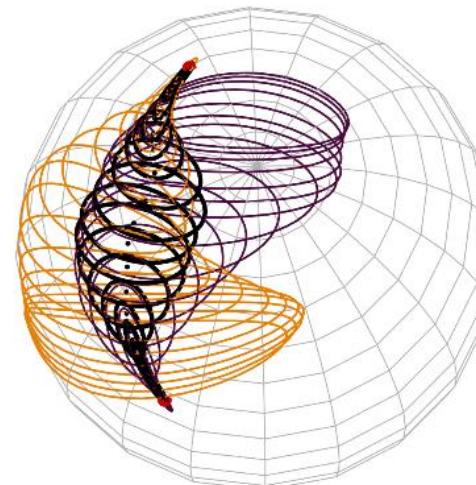
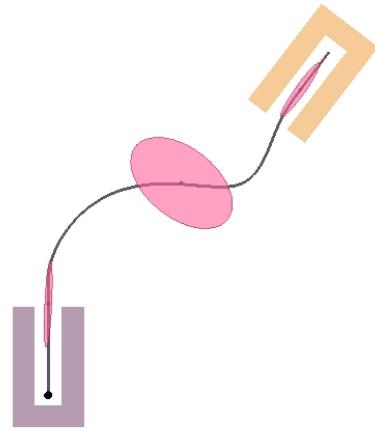
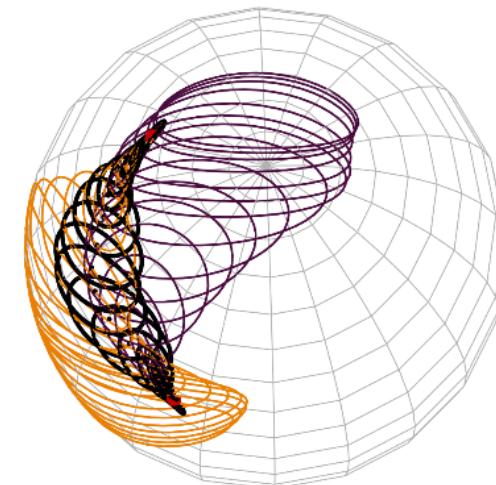
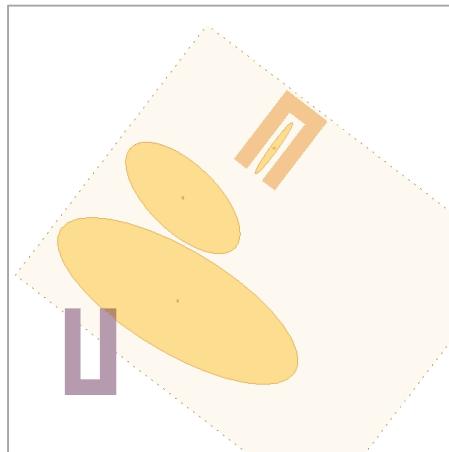
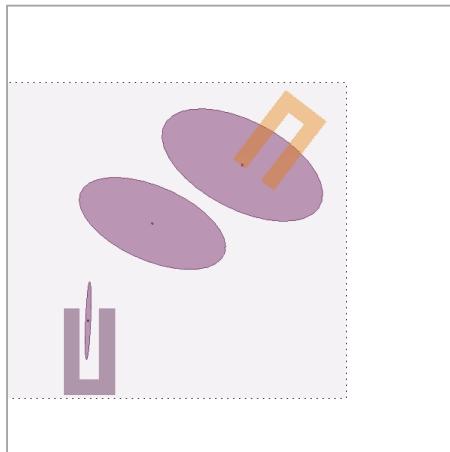
[Jaquier and Calinon, IROS 2017]

Fusion (product of Gaussians) on manifolds



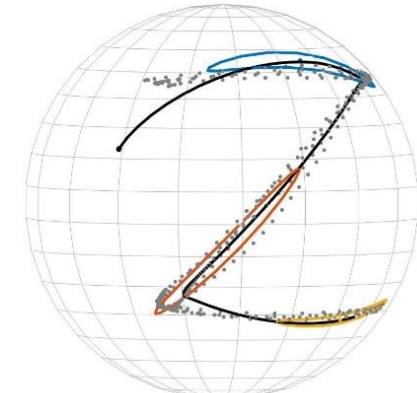
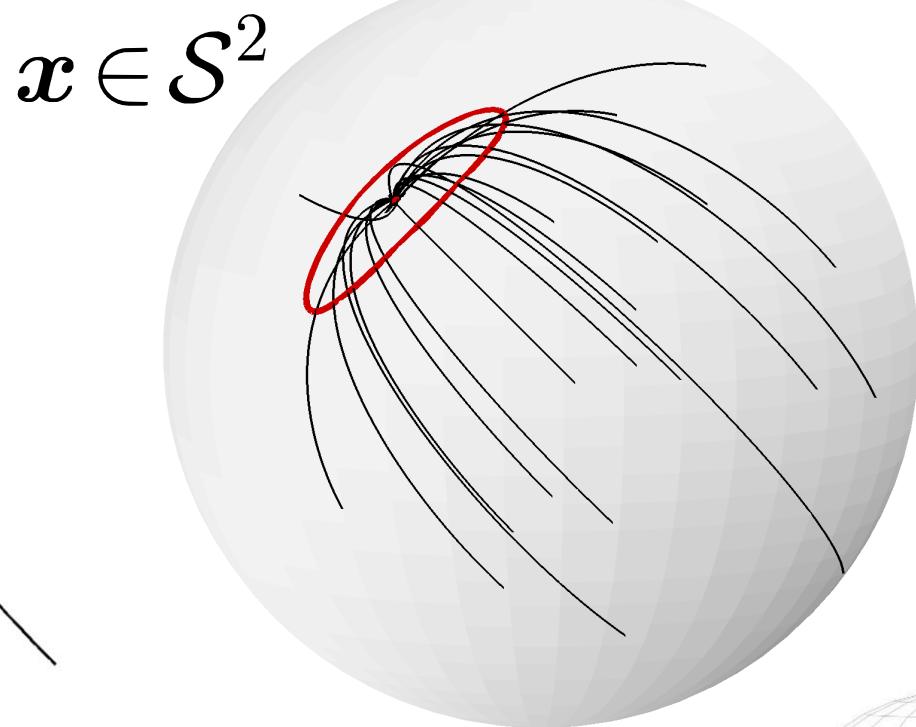
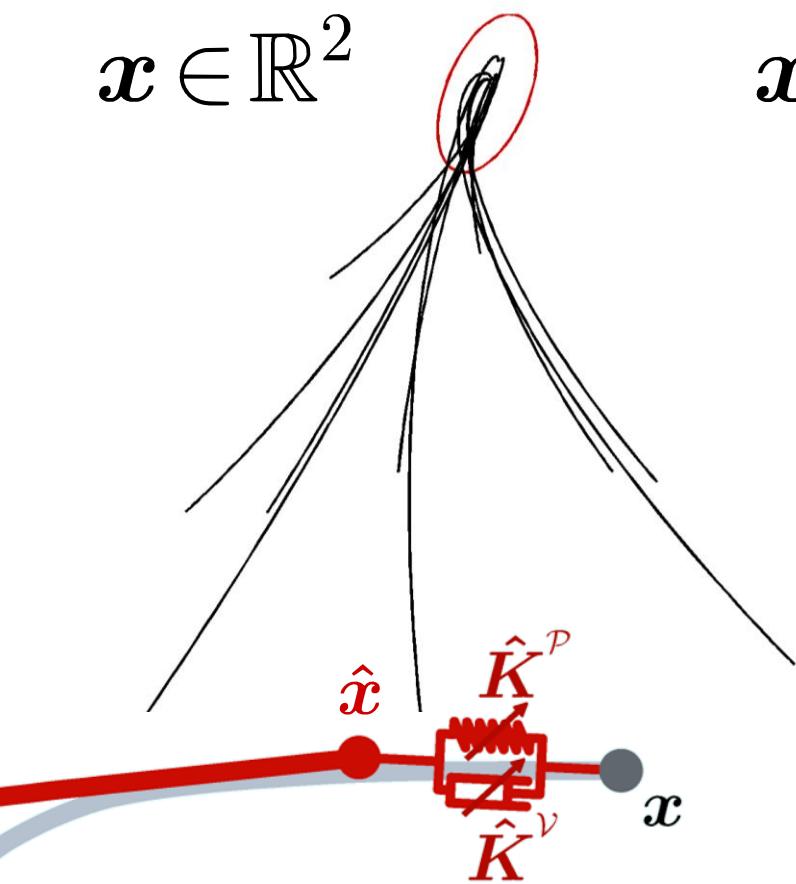
K Gaussians

Fusion (product of Gaussians) on manifolds



→ Control the orientation of the robot hand in accordance to the orientation of objects, tools or virtual landmarks

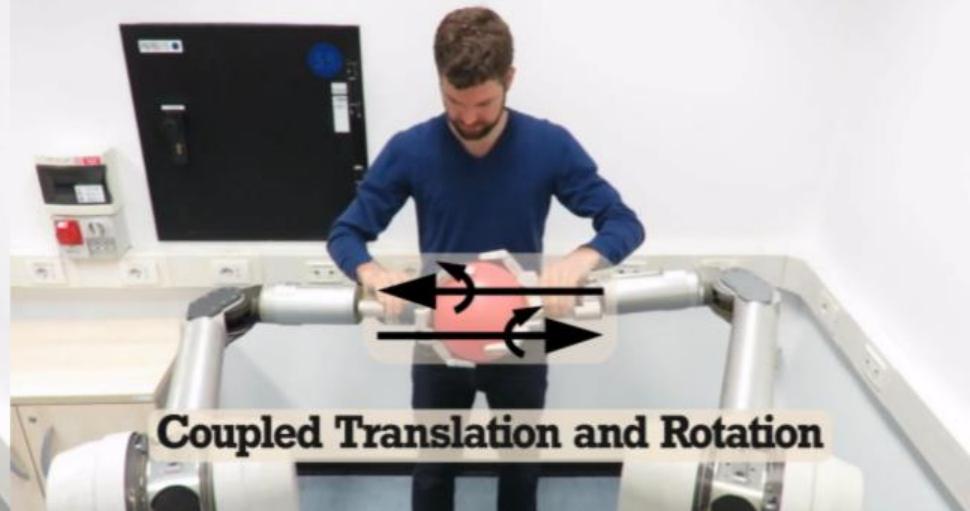
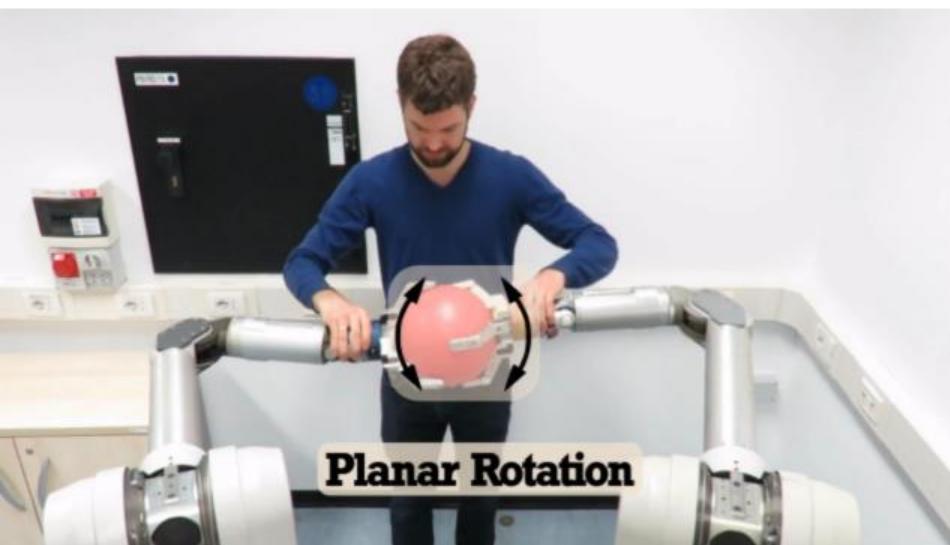
Control on Riemannian manifolds



Example: Model predictive control of orientation with unit quaternions by using Riemannian geometry

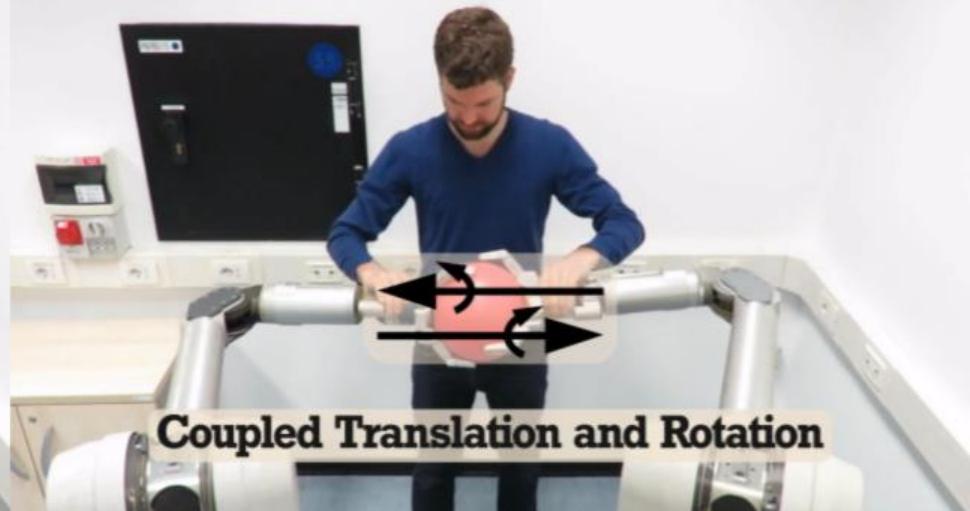
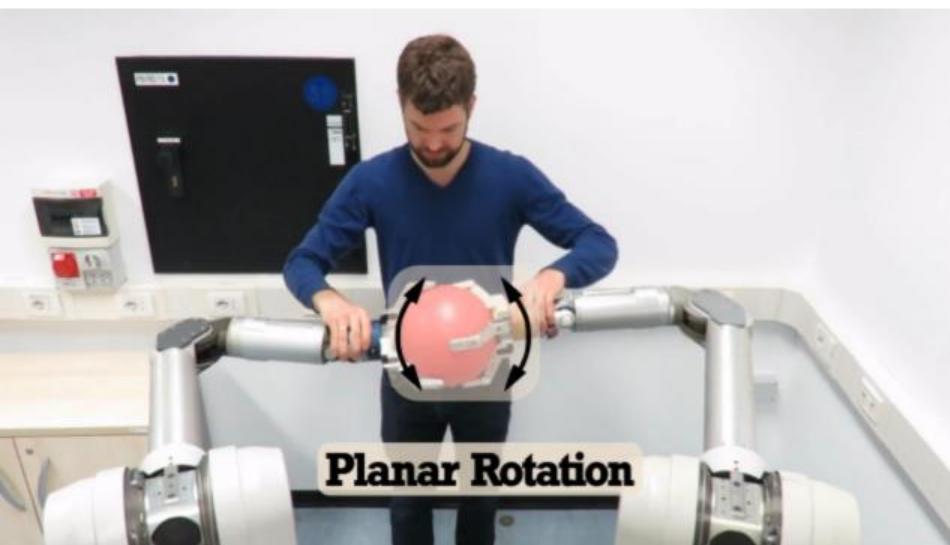
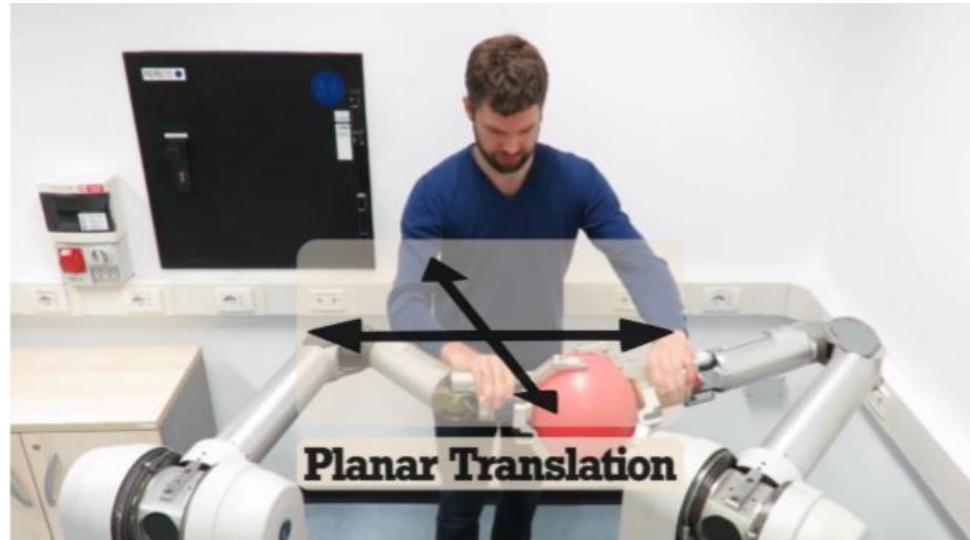
Control on Riemannian manifolds

We demonstrate three different tasks, each requiring a different synergy between the end-effectors.

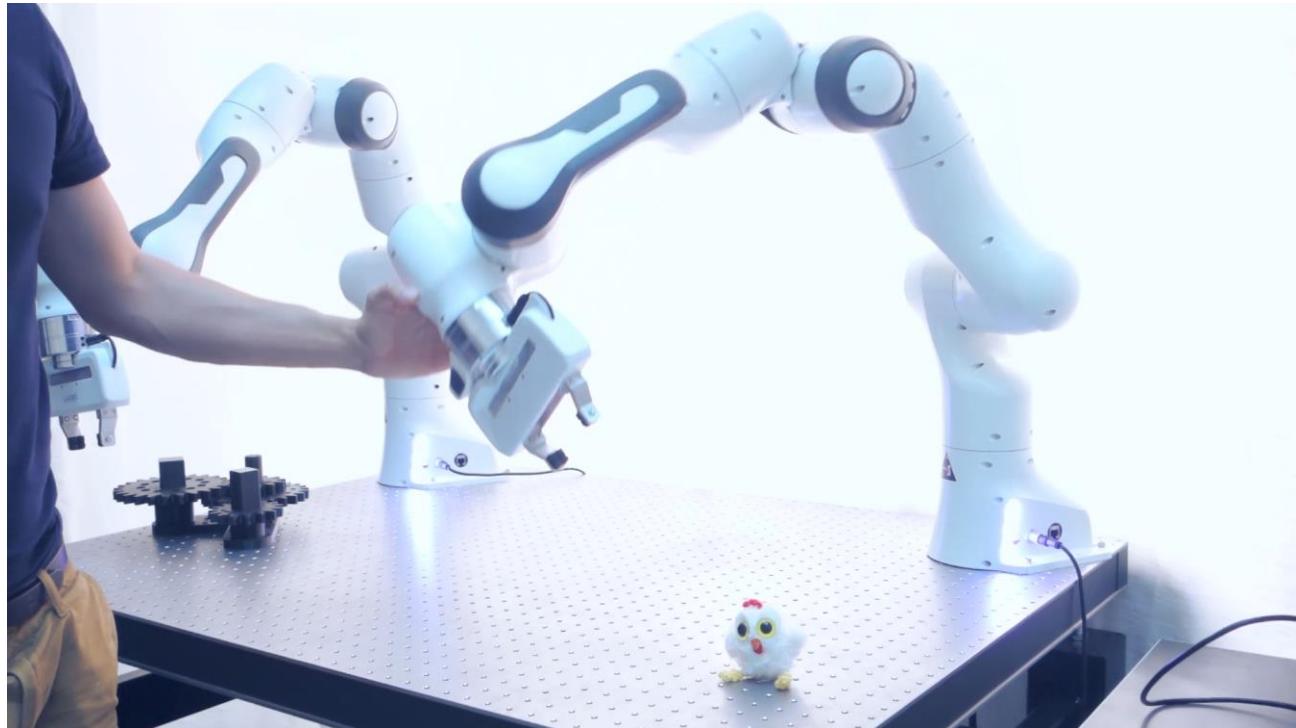
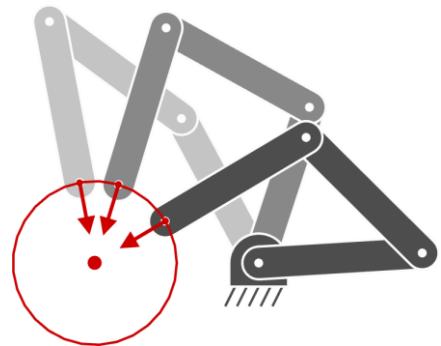
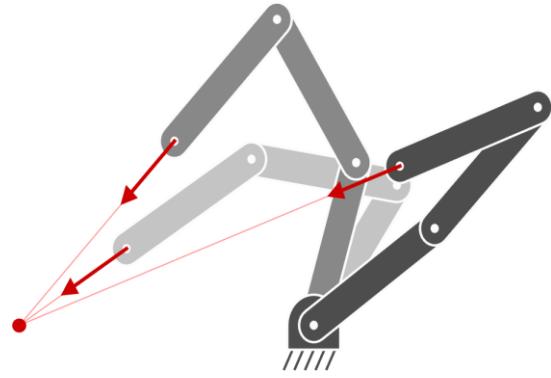
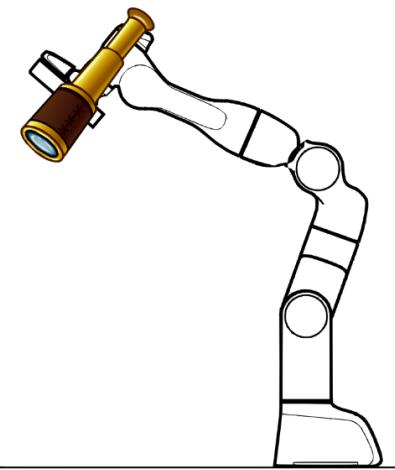
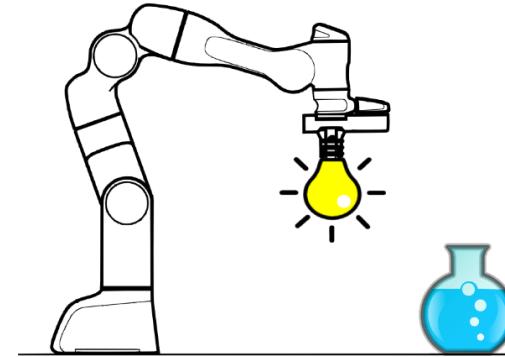
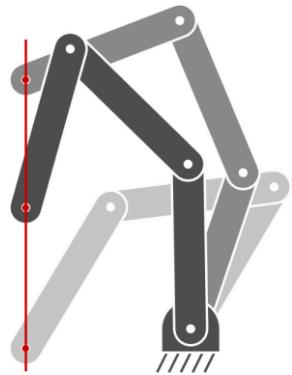
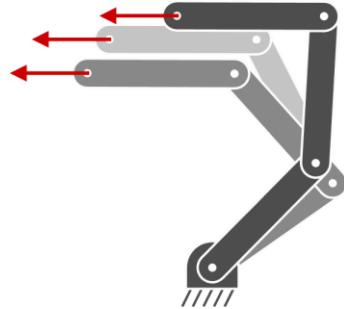


Control on Riemannian manifolds

We demonstrate three different tasks, each requiring a different synergy between the end-effectors.



Platform for reproducible research in science



Riemannian geometry - Resources

Softwares

<http://www.idiap.ch/software/pbdlib/>

Matlab codes: demo_Riemannian_sphere_GMM01.m

C++ codes: demo_Riemannian_sphere_GMM01.cpp

References

[Zeestraten, Havoutis, Silvério, Calinon and Caldwell, “*An Approach for Imitation Learning on Riemannian Manifolds*”, IEEE Robotics and Automation Letters 2(3), 2017]

[Jaquier and Calinon, “*Gaussian Mixture Regression on Symmetric Positive Definite Matrices Manifolds: Application to Wrist Motion Estimation with sEMG*”, IROS’2017]

Source codes

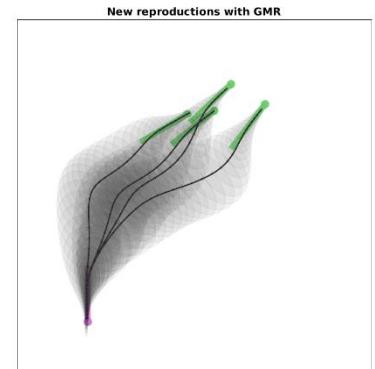
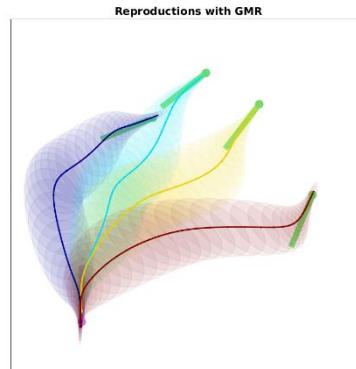
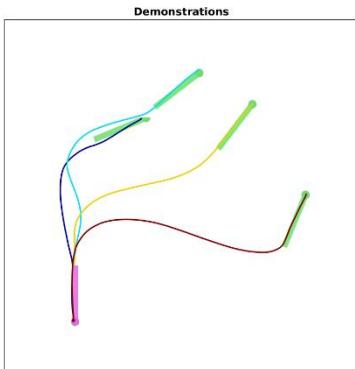
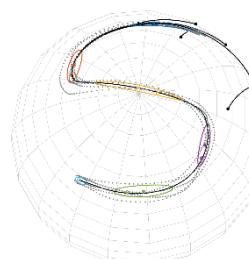
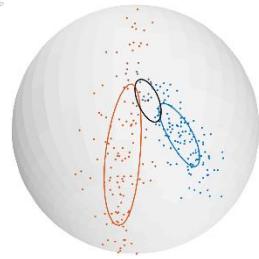
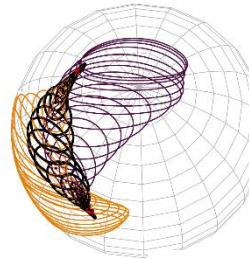
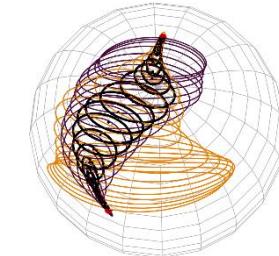
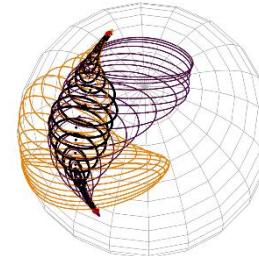
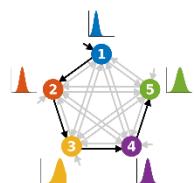
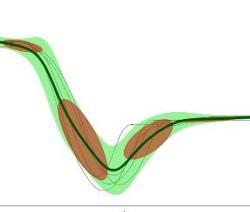
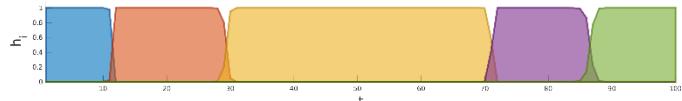
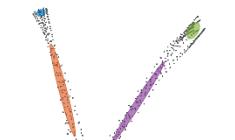
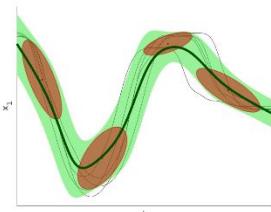
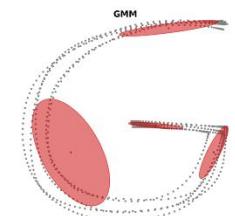
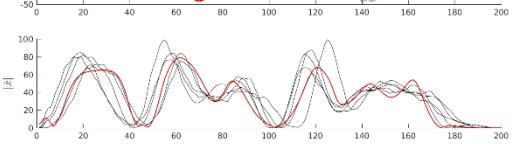
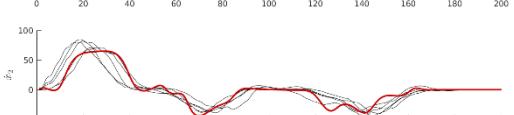
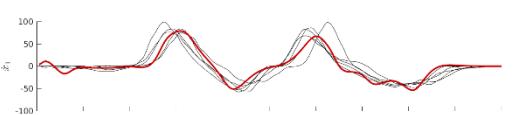
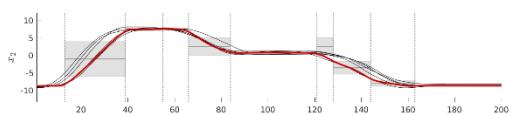
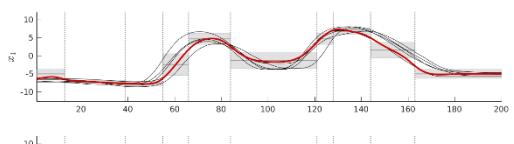
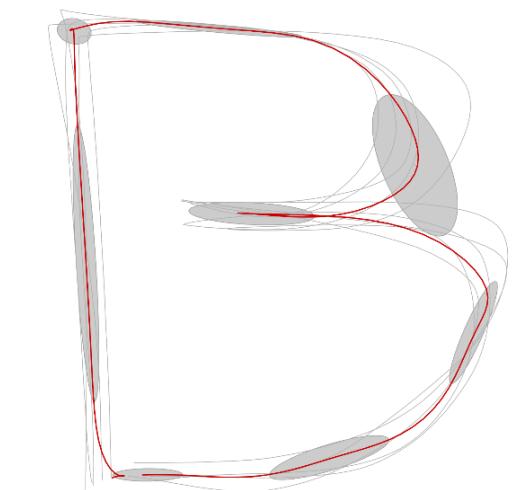
<http://www.idiap.ch/software/pbdlib/>

Matlab / GNU Octave

C++

Python

Pbdlib



<http://www.idiap.ch/software/pbdlib/>

PbDlib

PbDlib is a collection of source codes for robot programming by demonstration (learning from demonstration). It includes a varied set of functionalities at the crossroad of statistical learning, dynamical systems, optimal control and differential geometry. It is available in the following languages:

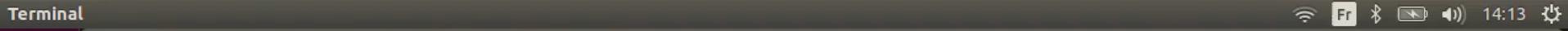
- Matlab / GNU Octave
- C++
- Python

PbDlib can be used in applications requiring task adaptation, human-robot skill transfer, safe controllers based on minimal intervention principle, as well as for probabilistic motion analysis and synthesis in multiple coordinate systems.

rli@pavilion01: ~/Documents/pbdlib-cpp/build

rli@pavilion01:~/Documents/pbdlib-cpp/build\$./demo_HSMM_batchLQR01

[



I

rli@pavilion01:~/Documents/pbdlib-cpp/build

rli@pavilion01:~/Documents/pbdlib-cpp/build\$./demo_TPGMR01

I

rli@pavilion01:~/Documents/pbdlib-cpp/build

rli@pavilion01:~/Documents/pbdlib-cpp/build\$./demo_TPbatchLQR01

I

Source codes (Matlab/Octave, C++ and Python):

<http://www.idiap.ch/software/pbdlib/>

Contact:

sylvain.calinon@idiap.ch

<http://calinon.ch>



Photo: Basilio Noris