Learning Algorithms for Robotics Systems

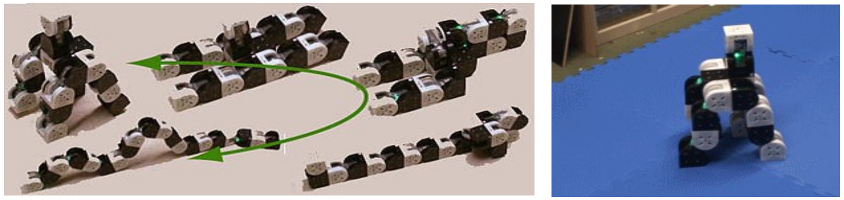
Ph.D. candidate: Alberto Dalla Libera
Advisor: Ruggero Carli
Co-Advisor: Gianluigi Pillonetto

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

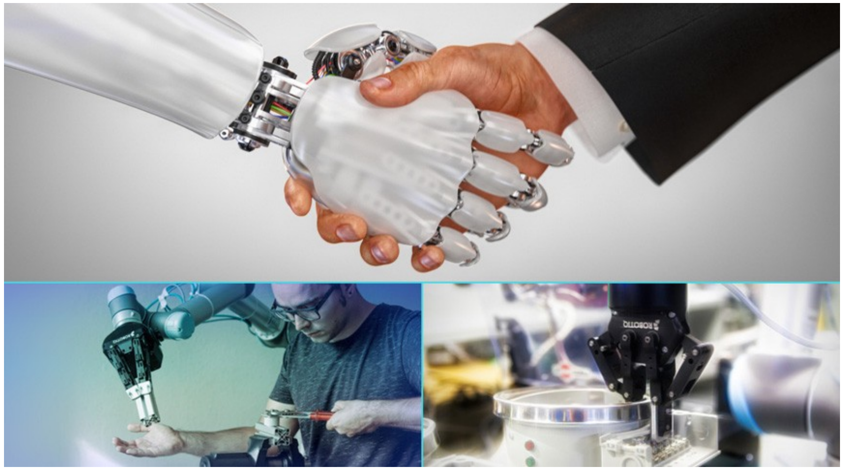
Motivation

Robotics systems are becoming always more and more **autonomous** and **reconfigurable**, as well as used in **“out of the cage” applications**:

MODULAR ROBOTICS
(M-TRAN 3, AIST JAPAN)



COLLABORATIVE ROBOTICS



SERVICE ROBOTS

Motivation

CHALLENGING ISSUES:

- Decreasing **manufacturing and set-up costs**
- **Limited prior knowledge** about the robot model:
  - **Kinematic** (e.g. Modular Robotics)
  - **Dynamics** (e.g. low-quality components)
- Unknown **external environment**:
  - Workspace, tools, and object to be manipulated
  - Human-robot interaction (e.g. Collaborative Robotics)
- Need of **autonomous algorithms** for control (e.g. set-up costs reduction)

POSITIVE ASPECT:

- Large availability of data (digital controller)

Motivation

CHALLENGING ISSUES:

- Decreasing **manufacturing and set-up costs**
- **Limited prior knowledge** about the robot model:
  - **Kinematic** (e.g. Modular Robotics)
  - **Dynamics** (e.g. low-quality components)
- Unknown **external environment**:
  - Workspace, tools, and object to be manipulated
  - Human-robot interaction (e.g. Collaborative Robotics)
- Need of **autonomous algorithms** for control (e.g. set-up costs reduction)

POSITIVE ASPECT:

- Large availability of data (digital controller)

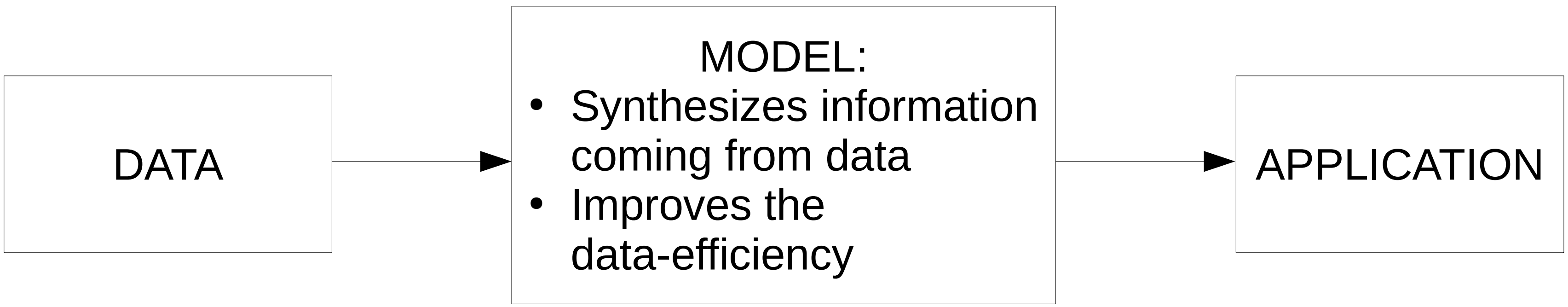
Can we face these challenges developing data-driven strategies?

Outline

- Motivation
- Thesis overview:
  - Autonomous learning of the robot Kinematics
  - Inverse dynamics identification: proprioceptive contact detection
  - Reinforcement Learning: MC-PILCO
- Geometrically Inspired Polynomial (GIP) kernel

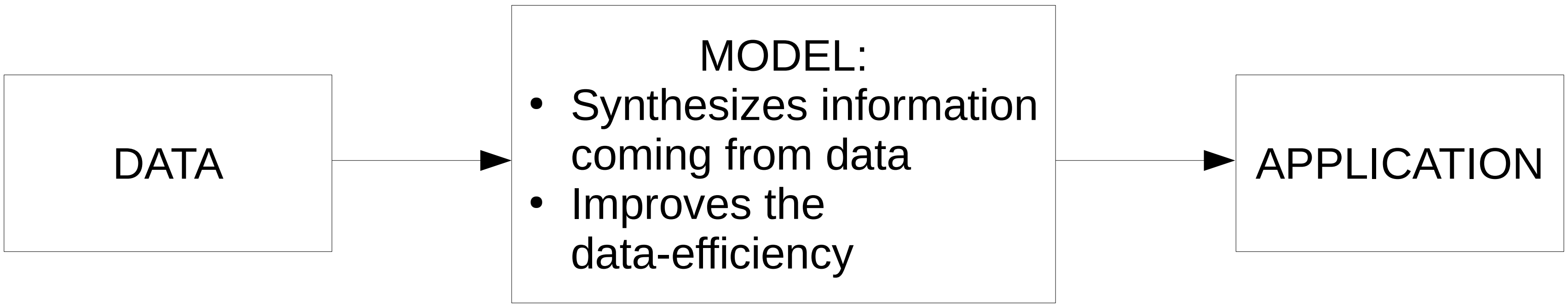
Thesis overview

General strategy adopted



Thesis overview

General strategy adopted



Problems considered

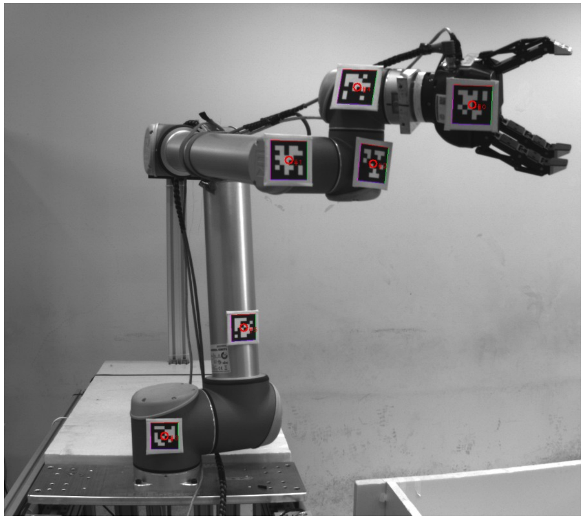
MODEL	APPLICATION
Kinematics	Kinematic controller
Inverse Dynamics	Proprioceptive contact detection
Forward Dynamics	Controller based on Reinforcement Learning (RL)

Autonomous learning of the Kinematics

NO PRIOR INFORMATION ABOUT THE KINEMATICS

SETUP:
- 2D camera
- Fiducial markers (one for each link)

MEASURES
- Joint values
- Marker poses



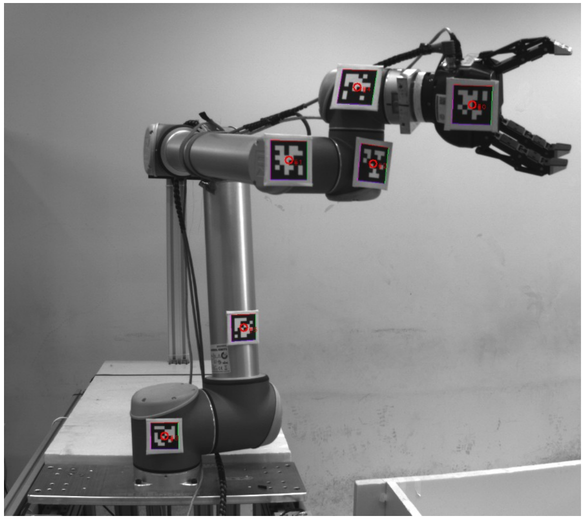
REFERENCES:
- 2019 18th European Control Conference (ECC), Naples, Italy, 2019, pp. 1586-1591.
  A. Dalla Libera, M. Terzi, A. Rossi, G. A. Susto, R. Carli, Robot kinematic structure classification from time series of visual data
- IEEE T-RO (submitted)
  A. Dalla Libera, N. Castama, S. Ghidoni, R. Carli, Autonomous learning of the robot kinematics structure

Autonomous learning of the Kinematics

NO PRIOR INFORMATION ABOUT THE KINEMATICS

SETUP:
- 2D camera
- Fiducial markers (one for each link)

MEASURES
- Joint values
- Marker poses



DATA →

KINEMATIC STRUCTURE CLASSIFICATION

Identification of:
- Joints order
- Joints type (prismatic or revolute)
- Markers/links order

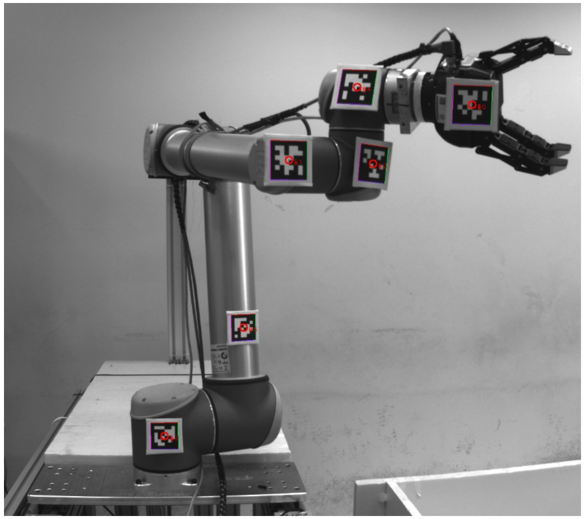
REFERENCES:
- 2019 18th European Control Conference (ECC), Naples, Italy, 2019, pp. 1586-1591.
  A. Dalla Libera, M. Terzi, A. Rossi, G. A. Susto, R. Carli, Robot kinematic structure classification from time series of visual data
- IEEE T-RO (submitted)
  A. Dalla Libera, N. Castama, S. Ghidoni, R. Carli, Autonomous learning of the robot kinematics structure

Autonomous learning of the Kinematics

NO PRIOR INFORMATION ABOUT THE KINEMATICS

SETUP:
- 2D camera
- Fiducial markers (one for each link)

MEASURES
- Joint values
- Marker poses



DATA →

KINEMATIC STRUCTURE CLASSIFICATION

Identification of:
- Joints order
- Joints type (prismatic or revolute)
- Markers/links order

DATA →

FORWARD KINEMATICS IDENTIFICATION:

$\hat{p}_M = \text{Pose of the last marker}$

$\boldsymbol{q} = \text{Joints coordinates}$

$\hat{p}_M = \hat{f}\left(\boldsymbol{q}\right)$

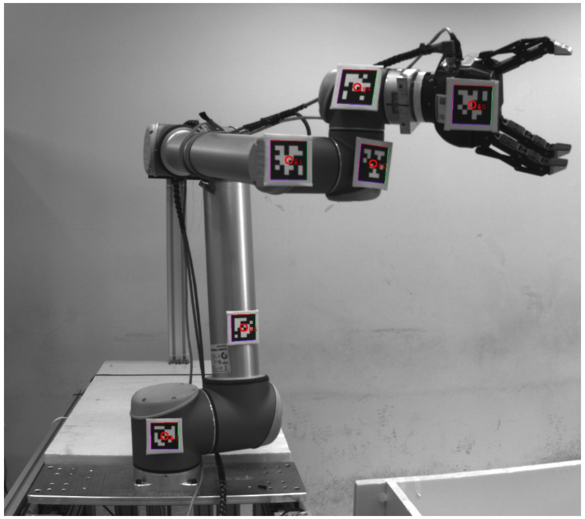
REFERENCES:
- 2019 18th European Control Conference (ECC), Naples, Italy, 2019, pp. 1586-1591.
  A. Dalla Libera, M. Terzi, A. Rossi, G. A. Susto, R. Carli, Robot kinematic structure classification from time series of visual data
- IEEE T-RO (submitted)
  A. Dalla Libera, N. Castama, S. Ghidoni, R. Carli, Autonomous learning of the robot kinematics structure

Autonomous learning of the Kinematics

NO PRIOR INFORMATION ABOUT THE KINEMATICS

SETUP:
- 2D camera
- Fiducial markers (one for each link)

MEASURES
- Joint values
- Marker poses



DATA → KINEMATIC STRUCTURE CLASSIFICATION

Identification of:
- Joints order
- Joints type (prismatic or revolute)
- Markers/links order

DATA → FORWARD KINEMATICS IDENTIFICATION:

$\hat{p}_M$ = Pose of the last marker

$\boldsymbol{q}$ = Joints coordinates

$\hat{p}_M = \hat{f}(\boldsymbol{q})$

Identified using Gaussian process regression

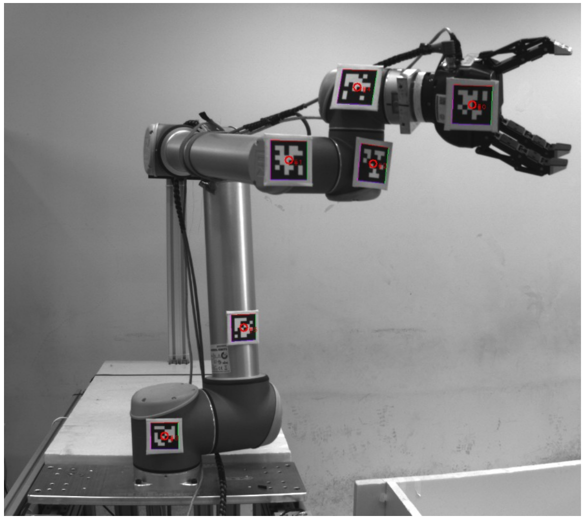
REFERENCES:
- 2019 18th European Control Conference (ECC), Naples, Italy, 2019, pp. 1586-1591.
  A. Dalla Libera, M. Terzi, A. Rossi, G. A. Susto, R. Carli, Robot kinematic structure classification from time series of visual data
- IEEE T-RO (submitted)
  A. Dalla Libera, N. Castama, S. Ghidoni, R. Carli, Autonomous learning of the robot kinematics structure

Autonomous learning of the Kinematics

NO PRIOR INFORMATION ABOUT THE KINEMATICS

SETUP:
- 2D camera
- Fiducial markers (one for each link)

MEASURES
- Joint values
- Marker poses



DATA → KINEMATIC STRUCTURE CLASSIFICATION

Identification of:
- Joints order
- Joints type (prismatic or revolute)
- Markers/links order

DATA → FORWARD KINEMATICS IDENTIFICATION:

$\hat{p}_M$ = Pose of the last marker

$\boldsymbol{q}$ = Joints coordinates

$\hat{p}_M = \hat{f}(\boldsymbol{q})$ — Identified using Gaussian process regression

KINEMATICS CONTROLLER:

T = Sampling time

$\boldsymbol{p}_*$ = Target pose

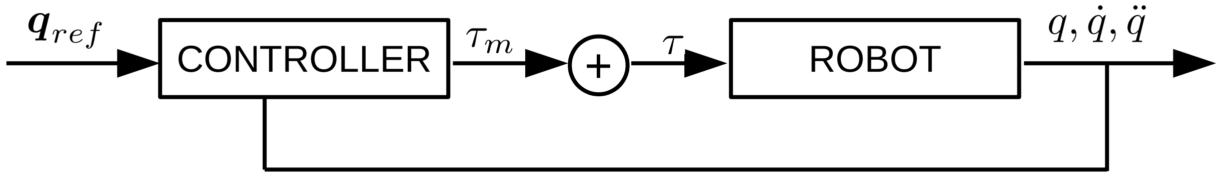
$\boldsymbol{q}_d$ = Desired joint configuration

$\boldsymbol{q}_d(t+T) = \boldsymbol{q}_d + \gamma \nabla_{\boldsymbol{q}} (p_* - \hat{p}_M)$

REFERENCES:
- 2019 18th European Control Conference (ECC), Naples, Italy, 2019, pp. 1586-1591.
  A. Dalla Libera, M. Terzi, A. Rossi, G. A. Susto, R. Carli, Robot kinematic structure classification from time series of visual data
- IEEE T-RO (submitted)
  A. Dalla Libera, N. Castama, S. Ghidoni, R. Carli, Autonomous learning of the robot kinematics structure

Proprioceptive contact detection

TASK: detect contacts using only proprioceptive measures (torques and joint coordinates)



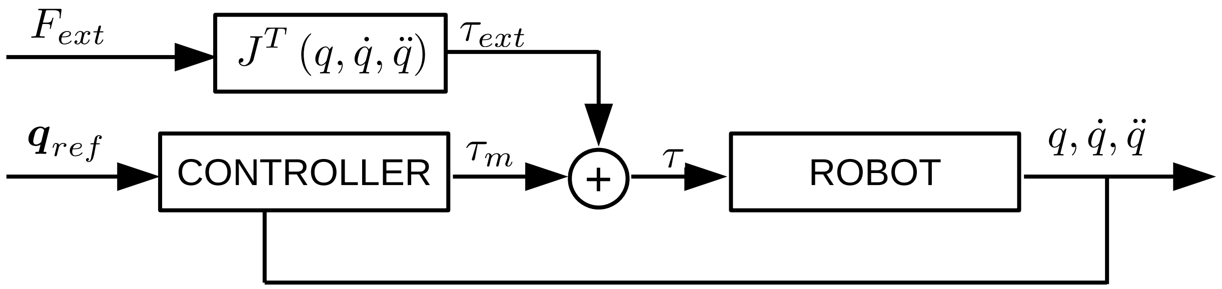
REFERENCE:

- 2019 American Control Conference (ACC), Philadelphia, PA, USA, 2019, pp. 19-24.
  A. Dalla Libera, E. Tosello, G. Pillonetto, S. Ghidoni, R. Carli, Proprioceptive Robot Collision Detection through Gaussian Process Regression

Proprioceptive contact detection

TASK: detect contacts using only proprioceptive measures (torques and joint coordinates)

When considering external forces...

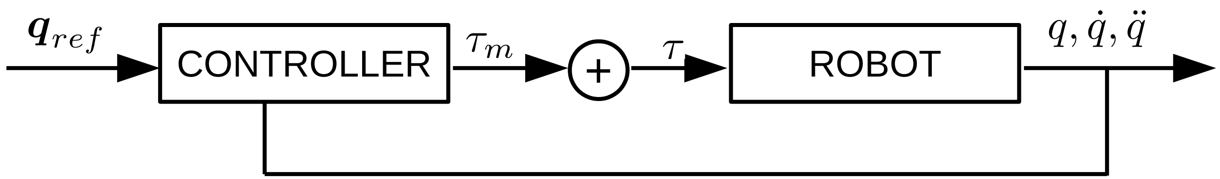


REFERENCE:

- 2019 American Control Conference (ACC), Philadelphia, PA, USA, 2019, pp. 19-24.
  A. Dalla Libera, E. Tosello, G. Pillonetto, S. Ghidoni, R. Carli, Proprioceptive Robot Collision Detection through Gaussian Process Regression

Proprioceptive contact detection

TASK: detect contacts using only proprioceptive measures (torques and joint coordinates)



DIRECT ESTIMATION ALGORITHM:

- Derive an **inverse dynamics estimator**

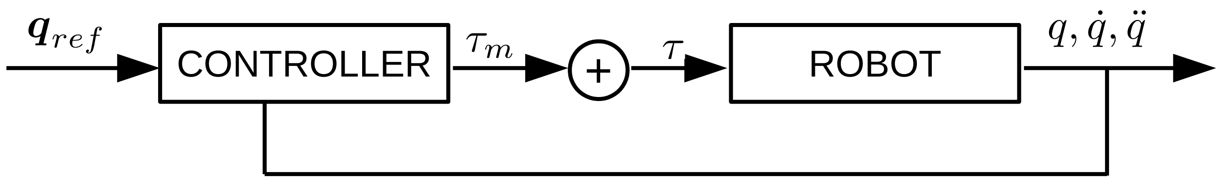
$$X = \{(q_1, \dot{q}_1, \ddot{q}_1), \ldots, (q_N, \dot{q}_N, \ddot{q}_N)\}$$
$$Y = \{\tau_{m_1}, \ldots, \tau_{m_N}\}$$
$$\Downarrow$$
$$\hat{\tau}_m(q, \dot{q}, \ddot{q})$$

REFERENCE:

- 2019 American Control Conference (ACC), Philadelphia, PA, USA, 2019, pp. 19-24.
  A. Dalla Libera, E. Tosello, G. Pillonetto, S. Ghidoni, R. Carli, Proprioceptive Robot Collision Detection through Gaussian Process Regression

Proprioceptive contact detection

TASK: detect contacts using only proprioceptive measures (torques and joint coordinates)



DIRECT ESTIMATION ALGORITHM:

- Derive an **inverse dynamics estimator**
- Characterize the estimator accuracy defining a **threshold** (e.g. **max error in a test set**)

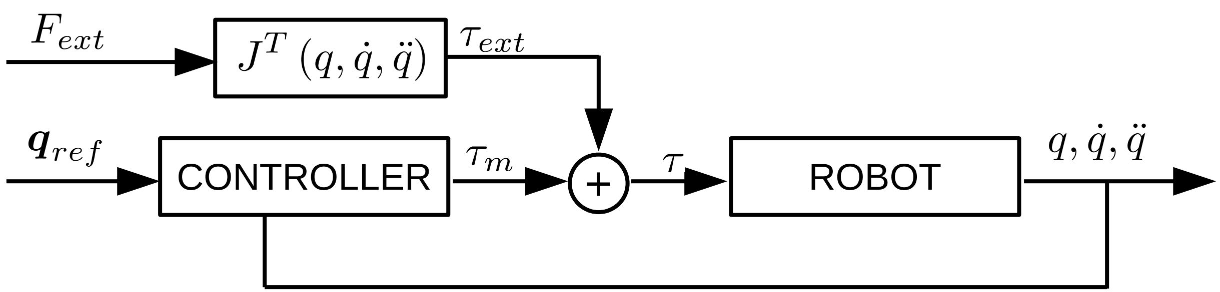
$$X = \{(q_1, \dot{q}_1, \ddot{q}_1), \dots, (q_N, \dot{q}_N, \ddot{q}_N)\}$$
$$Y = \{\tau_{m_1}, \dots, \tau_{m_N}\}$$
$$\Downarrow$$
$$\hat{\tau}_m(q, \dot{q}, \ddot{q})$$

$$e_\tau = |\tau_m - \hat{\tau}_m|$$
$$\sigma_{CD} = max(e_\tau)$$

REFERENCE:
- 2019 American Control Conference (ACC), Philadelphia, PA, USA, 2019, pp. 19-24.
  A. Dalla Libera, E. Tosello, G. Pillonetto, S. Ghidoni, R. Carli, Proprioceptive Robot Collision Detection through Gaussian Process Regression

Proprioceptive contact detection

TASK: detect contacts using only proprioceptive measures (torques and joint coordinates)



DIRECT ESTIMATION ALGORITHM:

- Derive an **inverse dynamics estimator**

$$X = \{(q_1,\dot{q}_1,\ddot{q}_1),\dots,(q_N,\dot{q}_N,\ddot{q}_N)\}$$
$$Y = \{\tau_{m_1},\dots,\tau_{m_N}\}$$
$$\hat{\tau}_m(q,\dot{q},\ddot{q})$$

- Characterize the estimator accuracy defining a **threshold** (e.g. **max error in a test set)**

$$e_\tau = |\tau_m - \hat{\tau}_m|$$
$$\sigma_{CD} = max(e_\tau)$$

- A collision occurred if the **estimation error is greater than the threshold**

$$f_{CD}(e_\tau) = \begin{cases} \text{TRUE}, & if\ |e_\tau| \geq \boldsymbol{\sigma_{CD}} \\ \text{FALSE}, & if\ |e_\tau| < \boldsymbol{\sigma_{CD}} \end{cases}$$

REFERENCE:

- 2019 American Control Conference (ACC), Philadelphia, PA, USA, 2019, pp. 19-24.
  A. Dalla Libera, E. Tosello, G. Pillonetto, S. Ghidoni, R. Carli, Proprioceptive Robot Collision Detection through Gaussian Process Regression

MC-PILCO

OBJECTIVE: learning to perform a task based on data acquired interacting with the system
MC-PILCO: Monte Carlo Probabilistic inference for learning Control

$$\boldsymbol{x}(t) = \text{System state at time } t$$

$$\boldsymbol{u}(t) = \text{Input vector at time } t$$

$$\pi\left(\boldsymbol{x}(t), \boldsymbol{\theta}\right) = \boldsymbol{u}\left(t\right) = \text{ Policy } (\boldsymbol{\theta} = \text{ Policy parameters})$$

$$c\left(\boldsymbol{x}(t)\right) \geq 0 = \text{ Local cost: encodes the task}$$

$$C\left(\boldsymbol{x}(0), \pi\right) = \sum_{t=0}^{T} c\left(\boldsymbol{x}(t)\right) = \text{ Cumulative cost}$$

THE GOAL IS MINIMIZING THE EXPECTED VALUE OF THE CUMULATIVE COST

MC-PILCO

OBJECTIVE: learning to perform a task based on data acquired interacting with the system
MC-PILCO: Monte Carlo Probabilistic inference for learning Control

$$\boldsymbol{x}(t) = \text{System state at time } t$$

$$\boldsymbol{u}(t) = \text{Input vector at time } t$$

$$\pi\left(\boldsymbol{x}(t), \boldsymbol{\theta}\right) = \boldsymbol{u}\left(t\right) = \text{ Policy } (\boldsymbol{\theta} = \text{ Policy parameters})$$

$$c\left(\boldsymbol{x}(t)\right) \geq 0 = \text{ Local cost: encodes the task}$$

$$C\left(\boldsymbol{x}(0), \pi\right) = \sum_{t=0}^{T} c\left(\boldsymbol{x}(t)\right) = \text{ Cumulative cost}$$

THE GOAL IS MINIMIZING THE EXPECTED VALUE OF THE CUMULATIVE COST

π_0 →

TEST ON THE REAL SYSTEM

$$\boldsymbol{x}(t) = \left[\boldsymbol{q}^T\left(t\right) \dot{\boldsymbol{q}}^T\left(t\right)\right]$$

$$\mathcal{D} = \left\{\langle \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{x}(t+1)\rangle_{t=0}^{T}\right\}$$

MC-PILCO

OBJECTIVE: learning to perform a task based on data acquired interacting with the system
MC-PILCO: Monte Carlo Probabilistic inference for learning Control

$\boldsymbol{x}(t) =$ System state at time t

$\boldsymbol{u}(t) =$ Input vector at time t

$\pi\left(\boldsymbol{x}(t), \boldsymbol{\theta}\right) = \boldsymbol{u}\left(t\right) =$ Policy ($\boldsymbol{\theta} =$ Policy parameters)

$c\left(\boldsymbol{x}(t)\right) \geq 0 =$ Local cost: encodes the task

$$C\left(\boldsymbol{x}(0), \pi\right) = \sum_{t=0}^{T} c\left(\boldsymbol{x}(t)\right) = \text{ Cumulative cost}$$

THE GOAL IS MINIMIZING THE EXPECTED VALUE OF THE CUMULATIVE COST

π_0 →

TEST ON THE REAL SYSTEM

$$\boldsymbol{x}(t) = \left[\boldsymbol{q}^T\left(t\right) \dot{\boldsymbol{q}}^T\left(t\right)\right]$$

$$\mathcal{D} = \left\{\langle \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{x}(t+1)\rangle_{t=0}^{T}\right\}$$

→

MODEL LEARNING: FORWARD DYNAMICS IDENTIFICATION

Prior model:

$$\boldsymbol{x}(t+1) = f\left(\boldsymbol{x}\left(t\right), \boldsymbol{u}\left(t\right)\right) + \boldsymbol{e}(t)$$

$f\left(\boldsymbol{x}\left(t\right), \boldsymbol{u}\left(t\right)\right)$ is a Gaussian process

Posterior:

$$\hat{\boldsymbol{x}}(t+1) \sim N\left(\hat{f}\left(\boldsymbol{x}\left(t\right), \boldsymbol{u}\left(t\right), \mathcal{D}\right), Var\left(\hat{f}\left(\boldsymbol{x}\left(t\right), \boldsymbol{u}\left(t\right)\right), \mathcal{D}\right)\right)$$

MC-PILCO

OBJECTIVE: learning to perform a task based on data acquired interacting with the system
MC-PILCO: Monte Carlo Probabilistic inference for learning Control

$\boldsymbol{x}(t) = \text{System state at time } t$

$\boldsymbol{u}(t) = \text{Input vector at time } t$

$\pi\left(\boldsymbol{x}(t), \boldsymbol{\theta}\right) = \boldsymbol{u}\left(t\right) = \text{ Policy } (\boldsymbol{\theta} = \text{ Policy parameters})$

$c\left(\boldsymbol{x}(t)\right) \geq 0 = \text{ Local cost: encodes the task}$

$$C\left(\boldsymbol{x}(0), \pi\right) = \sum_{t=0}^{T} c\left(\boldsymbol{x}(t)\right) = \text{ Cumulative cost}$$

THE GOAL IS MINIMIZING THE EXPECTED VALUE OF THE CUMULATIVE COST

π_0 →

TEST ON THE REAL SYSTEM

$$\boldsymbol{x}(t) = \left[\boldsymbol{q}^T\left(t\right) \dot{\boldsymbol{q}}^T\left(t\right)\right]$$

$$\mathcal{D} = \left\{\langle \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{x}(t+1)\rangle_{t=0}^{T}\right\}$$

→

MODEL LEARNING: FORWARD DYNAMICS IDENTIFICATION

Prior model:

$$\boldsymbol{x}(t+1) = f\left(\boldsymbol{x}\left(t\right), \boldsymbol{u}\left(t\right)\right) + \boldsymbol{e}(t)$$

$f\left(\boldsymbol{x}\left(t\right), \boldsymbol{u}\left(t\right)\right)$ is a Gaussian process

Posterior:

$$\hat{\boldsymbol{x}}(t+1) \sim N\left(\hat{f}\left(\boldsymbol{x}\left(t\right), \boldsymbol{u}\left(t\right), \mathcal{D}\right), Var\left(\hat{f}\left(\boldsymbol{x}\left(t\right), \boldsymbol{u}\left(t\right)\right), \mathcal{D}\right)\right)$$

POLICY LEARNING

$$\hat{\boldsymbol{\theta}} = \text{argmin}_{\theta} \hat{E}_{\hat{f}}\left[C\left(\boldsymbol{x}\left(0\right), \pi\right)\right]$$

$$\hat{E}_{\hat{f}}\left[C\left(\boldsymbol{x}\left(0\right), \pi\right)\right] = \text{Approximation of } \hat{E}\left[C\left(\boldsymbol{x}\left(0\right), \pi\right)\right]$$

Outline GIP kernel

Geometrically Inspired Polynomial (GIP) kernel:

- Background on inverse dynamics identification
  - Parametric approach
  - Non-parametric approach (Gaussian process regression)
- Derivation of the GIP kernel
- Numerical experiments

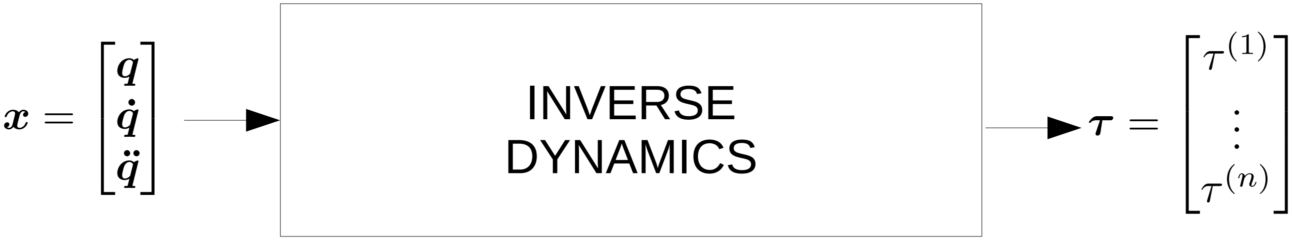
GIP kernel: background

INVERSE DYNAMICS: maps that relates joint trajectories and generalized torques

$n =$ Degrees of freedom

$\boldsymbol{q} =$ Joint coordinates

$\boldsymbol{\tau} =$ Generalized torques



$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \\ \ddot{\boldsymbol{q}} \end{bmatrix} \qquad \boldsymbol{\tau} = \begin{bmatrix} \tau^{(1)} \\ \vdots \\ \tau^{(n)} \end{bmatrix}$$

GIP kernel: background

INVERSE DYNAMICS: maps that relates joint trajectories and generalized torques

$n =$ Degrees of freedom

$\boldsymbol{q} =$ Joint coordinates

$\boldsymbol{\tau} =$ Generalized torques

$\boldsymbol{w}_d =$ Dynamics parameters

$\boldsymbol{w}_k =$ Kinematics parameters

$$\Phi\left(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}\right) = \begin{bmatrix} \phi^{(1)}\left(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{w}_k\right) \\ \vdots \\ \phi^{(n)}\left(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{w}_k\right) \end{bmatrix}$$

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \\ \ddot{\boldsymbol{q}} \end{bmatrix} \longrightarrow$$

Under rigid body assumptions:

$$\begin{aligned} \boldsymbol{\tau} &= M\left(\boldsymbol{q}\right)\ddot{\boldsymbol{q}} + C\left(\boldsymbol{q}, \dot{\boldsymbol{q}}\right)\dot{\boldsymbol{q}} + \boldsymbol{g}\left(\boldsymbol{q}\right) \\ &= \Phi\left(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{w}_k\right)\boldsymbol{w}_d \end{aligned}$$

$$\longrightarrow \boldsymbol{\tau} = \begin{bmatrix} \tau^{(1)} \\ \vdots \\ \tau^{(n)} \end{bmatrix}$$

GIP kernel: background

INVERSE DYNAMICS: maps that relates joint trajectories and generalized torques

$n =$ Degrees of freedom

$\boldsymbol{q} =$ Joint coordinates

$\boldsymbol{\tau} =$ Generalized torques

$\boldsymbol{w}_d =$ Dynamics parameters

$\boldsymbol{w}_k =$ Kinematics parameters

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \\ \ddot{\boldsymbol{q}} \end{bmatrix} \longrightarrow$$

Under rigid body assumptions:

$$\boldsymbol{\tau} = M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q})$$
$$= \Phi(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{w}_k)\boldsymbol{w}_d$$

$$\longrightarrow \boldsymbol{\tau} = \begin{bmatrix} \tau^{(1)} \\ \vdots \\ \tau^{(n)} \end{bmatrix}$$

Linear w.r.t. the dynamics parameters

$$\Phi(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \begin{bmatrix} \phi^{(1)}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{w}_k) \\ \vdots \\ \phi^{(n)}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{w}_k) \end{bmatrix}$$

Dependent on the kinematics parameters

GIP kernel: background

INVERSE DYNAMICS: maps that relates joint trajectories and generalized torques

$n =$ Degrees of freedom
$\boldsymbol{q} =$ Joint coordinates
$\boldsymbol{\tau} =$ Generalized torques
$\boldsymbol{w}_d =$ Dynamics parameters
$\boldsymbol{w}_k =$ Kinematics parameters

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \\ \ddot{\boldsymbol{q}} \end{bmatrix} \longrightarrow$$

Under rigid body assumptions:

$$\boldsymbol{\tau} = M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q})$$
$$= \Phi(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{w}_k)\boldsymbol{w}_d$$

$$\longrightarrow \boldsymbol{\tau} = \begin{bmatrix} \tau^{(1)} \\ \vdots \\ \tau^{(n)} \end{bmatrix}$$

Linear w.r.t. the dynamics parameters

$$\Phi(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \begin{bmatrix} \phi^{(1)}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{w}_k) \\ \vdots \\ \phi^{(n)}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{w}_k) \end{bmatrix}$$

Dependent on the kinematics parameters

PARAMETRIC IDENTIFICATION:

Training dataset
$X =$ Input locations $= \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_T\}$ $Y =$ Measures of $\boldsymbol{\tau}$ $= \{\boldsymbol{y}_1, \dots, \boldsymbol{y}_T\}$

Assuming the kinematics parameters known

$$\hat{\boldsymbol{w}} = \operatorname{argmin}_{\boldsymbol{w}} \sum_{t=1}^{T} ||\boldsymbol{y}_t - \Phi(\boldsymbol{x}_t)\boldsymbol{w}||^2$$

GIP kernel: background

- Deriving physical models requires effort
- Kinematics parameters could be unknown or partially known
- Uncertainty in the kinematics parameters
- Unmodeled behaviors like frictions, elasticity, and backlash

Non-parametric approach: GAUSSIAN PROCESS REGRESSION

GIP kernel: background

- Deriving physical models requires effort
- Kinematics parameters could be unknown or partially known
- Uncertainty in the kinematics parameters
- Unmodeled behaviors like frictions, elasticity, and backlash

↓

Non-parametric approach: GAUSSIAN PROCESS REGRESSION

	TRAINING	TEST
DATA:	$X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$	$X_* = \{\boldsymbol{x}_{1_*}, \ldots, \boldsymbol{x}_{T_*}\}$
	$\boldsymbol{y}^{(k)} = \left[y_1^{(k)}, \ldots, y_N^{(k)}\right]^T$	$\boldsymbol{y}_*^{(k)} = \left[y_{1_*}^{(k)}, \ldots, y_{N_*}^{(k)}\right]^T$

- Joint torques are assumed independent (given the inputs)

GIP kernel: background

- Deriving physical models requires effort
- Kinematics parameters could be unknown or partially known
- Uncertainty in the kinematics parameters
- Unmodeled behaviors like frictions, elasticity, and backlash

Non-parametric approach: GAUSSIAN PROCESS REGRESSION

	TRAINING	TEST
DATA:	$X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$	$X_* = \{\boldsymbol{x}_{1_*}, \ldots, \boldsymbol{x}_{T_*}\}$
	$\boldsymbol{y}^{(k)} = \left[y_1^{(k)}, \ldots, y_N^{(k)}\right]^T$	$\boldsymbol{y}_*^{(k)} = \left[y_{1_*}^{(k)}, \ldots, y_{N_*}^{(k)}\right]^T$

MODEL: $\begin{bmatrix} \boldsymbol{y}^{(k)} \\ \boldsymbol{y}_*^{(k)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}(X) \\ \boldsymbol{f}(X_*) \end{bmatrix} + \begin{bmatrix} \boldsymbol{e}(X) \\ \boldsymbol{e}(X_*) \end{bmatrix}$ with $\boldsymbol{e}(\cdot) \sim N\left(0, \sigma_e^2 I\right)$

- Joint torques are assumed independent (given the inputs)

GIP kernel: background

- Deriving physical models requires effort
- Kinematics parameters could be unknown or partially known
- Uncertainty in the kinematics parameters
- Unmodeled behaviors like frictions, elasticity, and backlash

Non-parametric approach: GAUSSIAN PROCESS REGRESSION

	TRAINING	TEST
DATA:	$X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_T\}$	$X_* = \{\boldsymbol{x}_{1_*}, \dots, \boldsymbol{x}_{T_*}\}$
	$\boldsymbol{y}^{(k)} = \left[y_1^{(k)}, \dots, y_N^{(k)}\right]^T$	$\boldsymbol{y}_*^{(k)} = \left[y_{1_*}^{(k)}, \dots, y_{N_*}^{(k)}\right]^T$

MODEL: $$\begin{bmatrix} \boldsymbol{y}^{(k)} \\ \boldsymbol{y}_*^{(k)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}(X) \\ \boldsymbol{f}(X_*) \end{bmatrix} + \begin{bmatrix} \boldsymbol{e}(X) \\ \boldsymbol{e}(X_*) \end{bmatrix} \text{ with } \boldsymbol{e}(\cdot) \sim N\left(0, \sigma_e^2 I\right)$$

PRIOR: $$\begin{bmatrix} \boldsymbol{f}(X) \\ \boldsymbol{f}(X_*) \end{bmatrix} \sim N\left(\begin{bmatrix} \boldsymbol{m_f}(X) \\ \boldsymbol{m_f}(X_*) \end{bmatrix}, \begin{bmatrix} K_f(X,X) & K_f(X,X_*) \\ K_f(X_*,X) & K_f(X_*,X_*) \end{bmatrix}\right)$$

- Joint torques are assumed independent (given the inputs)
- The kernel function defines the prior covariance:

$$K_f(X,X) = E[X,X] \in \mathbb{R}^{T \times T}$$

$$E\left[y_i^{(k)}, y_j^{(k)}\right] = k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$$

GIP kernel: background

- Deriving physical models requires effort
- Kinematics parameters could be unknown or partially known
- Uncertainty in the kinematics parameters
- Unmodeled behaviors like frictions, elasticity, and backlash

Non-parametric approach: GAUSSIAN PROCESS REGRESSION

	TRAINING	TEST
DATA:	$X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$	$X_* = \{\boldsymbol{x}_{1_*}, \ldots, \boldsymbol{x}_{T_*}\}$
	$\boldsymbol{y}^{(k)} = \left[y_1^{(k)}, \ldots, y_N^{(k)}\right]^T$	$\boldsymbol{y}_*^{(k)} = \left[y_{1_*}^{(k)}, \ldots, y_{N_*}^{(k)}\right]^T$

MODEL: $\begin{bmatrix} \boldsymbol{y}^{(k)} \\ \boldsymbol{y}_*^{(k)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}(X) \\ \boldsymbol{f}(X_*) \end{bmatrix} + \begin{bmatrix} \boldsymbol{e}(X) \\ \boldsymbol{e}(X_*) \end{bmatrix}$ with $\boldsymbol{e}(\cdot) \sim N\left(0, \sigma_e^2 I\right)$

PRIOR: $\begin{bmatrix} \boldsymbol{f}(X) \\ \boldsymbol{f}(X_*) \end{bmatrix} \sim N\left(\begin{bmatrix} \boldsymbol{m}_{\boldsymbol{f}}(X) \\ \boldsymbol{m}_{\boldsymbol{f}}(X_*) \end{bmatrix}, \begin{bmatrix} K_f(X, X) & K_f(X, X_*) \\ K_f(X_*, X) & K_f(X_*, X_*) \end{bmatrix}\right)$

POSTERIOR: $\hat{\boldsymbol{f}}(X_*) = K_f\left(X_*, X\right)\left(K_f(X, X) + \sigma_e^2 I\right)^{-1} \boldsymbol{y}^{(k)}$

- Joint torques are assumed independent (given the inputs)
- The kernel function defines the prior covariance:

  $K_f(X, X) = E\left[X, X\right] \in \mathbb{R}^{T \times T}$

  $E\left[y_i^{(k)}, y_j^{(k)}\right] = k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$
- The posterior can be computed in closed form

GIP kernel: background

It is a common considering the **mean null**, and focusing on the **the kernel** function:

MODEL BASED (MB)		
$$k_{\phi}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \left(\boldsymbol{\phi}^{(k)}\left(\boldsymbol{x}_i\right)\right)^T \Sigma_{\phi} \boldsymbol{\phi}^{(k)}\left(\boldsymbol{x}_j\right)$$	• Data-efficiency • Generalization	• Requires a model • Model bias • Unmodeled behaviors

GIP kernel: background

It is a common considering the **mean null**, and focusing on the **the kernel** function:

MODEL BASED (MB) $k_\phi\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \left(\boldsymbol{\phi}^{(k)}\left(\boldsymbol{x}_i\right)\right)^T \Sigma_\phi \boldsymbol{\phi}^{(k)}\left(\boldsymbol{x}_j\right)$	• Data-efficiency • Generalization	• Requires a model • Model bias • Unmodeled behaviors
NON-PARAMETRIC (NP) $k_{RBF}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \lambda e^{-\frac{\|\|\boldsymbol{x}_i - \boldsymbol{x}_j\|\|^2_\Sigma}{2}}$	• High model capacity • Good asymptotic performance • No prior information needed and no bias	• Low data-efficiency • Low generalization

GIP kernel: background

It is a common considering the **mean null**, and focusing on the **the kernel** function:

MODEL BASED (MB) $k_\phi(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left(\boldsymbol{\phi}^{(k)}(\boldsymbol{x}_i)\right)^T \Sigma_\phi \boldsymbol{\phi}^{(k)}(\boldsymbol{x}_j)$	• Data-efficiency • Generalization	• Requires a model • Model bias • Unmodeled behaviors
NON-PARAMETRIC (NP) $k_{RBF}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \lambda e^{-\frac{\|\|\boldsymbol{x}_i - \boldsymbol{x}_j\|\|_\Sigma^2}{2}}$	• High model capacity • Good asymptotic performance • No prior information needed and no bias	• Low data-efficiency • Low generalization
SEMI-PARAMETRIC (SP) $k_{SP}(\boldsymbol{x}_i, \boldsymbol{x}_j) = k_\phi(\boldsymbol{x}_i, \boldsymbol{x}_j) + k_{RBF}(\boldsymbol{x}_i, \boldsymbol{x}_j)$	Merges the strengths of the two approaches: • MB => generalization • NP => accuracy	• Requires a model • NP compensation could be local

GIP kernel: background

It is a common considering the **mean null**, and focusing on the **the kernel** function:

MODEL BASED (MB) $k_\phi(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left(\boldsymbol{\phi}^{(k)}(\boldsymbol{x}_i)\right)^T \Sigma_\phi \boldsymbol{\phi}^{(k)}(\boldsymbol{x}_j)$	• Data-efficiency • Generalization	• Requires a model • Model bias • Unmodeled behaviors
NON-PARAMETRIC (NP) $k_{RBF}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \lambda e^{-\frac{\|\|\boldsymbol{x}_i - \boldsymbol{x}_j\|\|^2_\Sigma}{2}}$	• High model capacity • Good asymptotic performance • No prior information needed and no bias	• Low data-efficiency • Low generalization
SEMI-PARAMETRIC (SP) $k_{SP}(\boldsymbol{x}_i, \boldsymbol{x}_j) = k_\phi(\boldsymbol{x}_i, \boldsymbol{x}_j) + k_{RBF}(\boldsymbol{x}_i, \boldsymbol{x}_j)$	Merges the strengths of the two approaches: • MB => generalization • NP => accuracy	• Requires a model • NP compensation could be local

We aim at deriving a kernel with the following properties:

- No need of strong prior information
- Data-efficiency
- Good generalization
- Good asymptotic performance

GIP kernel: Derivation

POLYNOMIAL NOTATION

$\mathbb{P}_{[p]}\left(\boldsymbol{a}_{[p_a]}, \boldsymbol{b}_{[p_b]}\right) =$Polynomial functions with maximal degree p in the elements of $\boldsymbol{a}$ and $\boldsymbol{b}$ such that, for each monomial, the relative degrees of the elements of $\boldsymbol{a}$ (resp. $\boldsymbol{b}$) are $\leq p_a$ (resp. $\leq p_b$)

GIP kernel: Derivation

POLYNOMIAL NOTATION

$\mathbb{P}_{[p]}\left(\boldsymbol{a}_{[p_a]}, \boldsymbol{b}_{[p_b]}\right)$ =Polynomial functions with maximal degree p in the elements of $\boldsymbol{a}$ and $\boldsymbol{b}$ such that, for each monomial, the relative degrees of the elements of $\boldsymbol{a}$ (resp. $\boldsymbol{b}$) are $\leq p_a$ (resp. $\leq p_b$)

INPUT SPACE TRANSFORMATION

$$\boldsymbol{q}_c = \left[\cos\left(q^{(r_1)}\right) \dots \cos\left(q^{(r_{N_r})}\right)\right]^T \in \mathbb{R}^{N_r}$$

$$\boldsymbol{q}_s = \left[\sin\left(q^{(r_1)}\right) \dots \sin\left(q^{(r_{N_r})}\right)\right]^T \in \mathbb{R}^{N_r}$$

$$\boldsymbol{q}_{cs} = \left[\boldsymbol{q}_c^T \boldsymbol{q}_s^T\right]^T$$

$$\boldsymbol{q}_p = \left[q^{(p_1)} \dots q^{(p_{N_p})}\right]^T \in \mathbb{R}^{N_p}$$

$$\dot{\boldsymbol{q}}_v = Vec\left(\left\{\dot{q}^{(i)}\dot{q}^{(j)}\ ,\ 1 \leq i \leq n\ ,\ i \leq j \leq n\right\}\right)$$

$\mathcal{I}_r$ = indices of the revolute joints $= \{r_1, \dots, r_{N_r}\}$

$\mathcal{I}_p$ = indices of the prismatic joints $= \left\{p_1, \dots, r_{N_p}\right\}$

GIP kernel: Derivation

POLYNOMIAL NOTATION

$\mathbb{P}_{[p]}\left(\boldsymbol{a}_{[p_a]}, \boldsymbol{b}_{[p_b]}\right)$ =Polynomial functions with maximal degree p in the elements of $\boldsymbol{a}$ and $\boldsymbol{b}$ such that, for each monomial, the relative degrees of the elements of $\boldsymbol{a}$ (resp. $\boldsymbol{b}$) are $\leq p_a$ (resp. $\leq p_b$)

INPUT SPACE TRANSFORMATION

$$\boldsymbol{q}_c = \left[\cos\left(q^{(r_1)}\right) \dots \cos\left(q^{(r_{N_r})}\right)\right]^T \in \mathbb{R}^{N_r}$$

$$\boldsymbol{q}_s = \left[\sin\left(q^{(r_1)}\right) \dots \sin\left(q^{(r_{N_r})}\right)\right]^T \in \mathbb{R}^{N_r}$$

$$\boldsymbol{q}_{cs} = \left[\boldsymbol{q}_c^T \boldsymbol{q}_s^T\right]^T$$

$$\boldsymbol{q}_p = \left[q^{(p_1)} \dots q^{(p_{N_p})}\right]^T \in \mathbb{R}^{N_p}$$

$$\dot{\boldsymbol{q}}_v = Vec\left(\left\{\dot{q}^{(i)}\dot{q}^{(j)} \ , 1 \leq i \leq n \ , i \leq j \leq n\right\}\right)$$

$$\mathcal{I}_r = \text{indices of the revolute joints} = \{r_1, \dots, r_{N_r}\}$$

$$\mathcal{I}_p = \text{indices of the prismatic joints} = \{p_1, \dots, r_{N_p}\}$$

PROPOSITION: Characterization of the inverse dynamics as a polynomial function

The inverse dynamics is a polynomial function in $\mathbb{P}_{(2n+1)}\left(\boldsymbol{q}_{c_{(2)}}, \boldsymbol{q}_{s_{(2)}}, \boldsymbol{q}_{p_{(2)}}, \dot{\boldsymbol{q}}_{v_{(1)}}, \ddot{\boldsymbol{q}}_{(1)}\right)$.

Moreover, $\forall$ monomial $deg\left(\boldsymbol{q}_c^{(i)}\right) + deg\left(\boldsymbol{q}_s^{(i)}\right) \leq 2$.

REFERENCE:

- IEEE Robotics and Automation Letters. PP. 1-1. 10.1109/LRA.2019.2945240.
  A. Dalla Libera, R. Carli. (2019). A Data-Efficient Geometrically Inspired Polynomial Kernel for Robot Inverse Dynamics.

GIP kernel: Derivation

KERNEL PROPERTIES (RKHS interpretation)

- The RKHS of the inhomogeneous Polynomia kernel is composed of all the monomials up to the polynomial degree
- The product of kernels is still a kernel, and its RKHS is given by the convolution of the RKHS of the two kernels

$$\hat{f} \text{ is the MAP estimator } \iff \hat{f} = \text{argmin}_{f \in \mathcal{H}} ||\boldsymbol{y}^{(k)} - \boldsymbol{f}(X)||^2 + \sigma_n^2 ||f||_{\mathcal{H}}^2 \text{ with } \mathcal{H} \text{ RKHS of } k$$

$$k_{pl(p)}\left(\boldsymbol{a}(t_h), \boldsymbol{a}(t_j)\right) \rightarrow \mathbb{P}_{[p]}\left(\boldsymbol{a}_{[p]}\right)$$

$$k_{pl(p_a)}\left(\boldsymbol{a}(t_h), \boldsymbol{a}(t_j)\right) k_{pl(p_b)}\left(\boldsymbol{b}(t_h), \boldsymbol{b}(t_j)\right) \rightarrow \mathbb{P}_{[p_a+p_b]}\left(\boldsymbol{a}_{[p_a]}, \boldsymbol{b}_{[p_b]}\right)$$

REFERENCE:

- IEEE Robotics and Automation Letters. PP. 1-1. 10.1109/LRA.2019.2945240.
  A. Dalla Libera, R. Carli. (2019). A Data-Efficient Geometrically Inspired Polynomial Kernel for Robot Inverse Dynamics.

GIP kernel: Derivation

KERNEL PROPERTIES (RKHS interpretation)

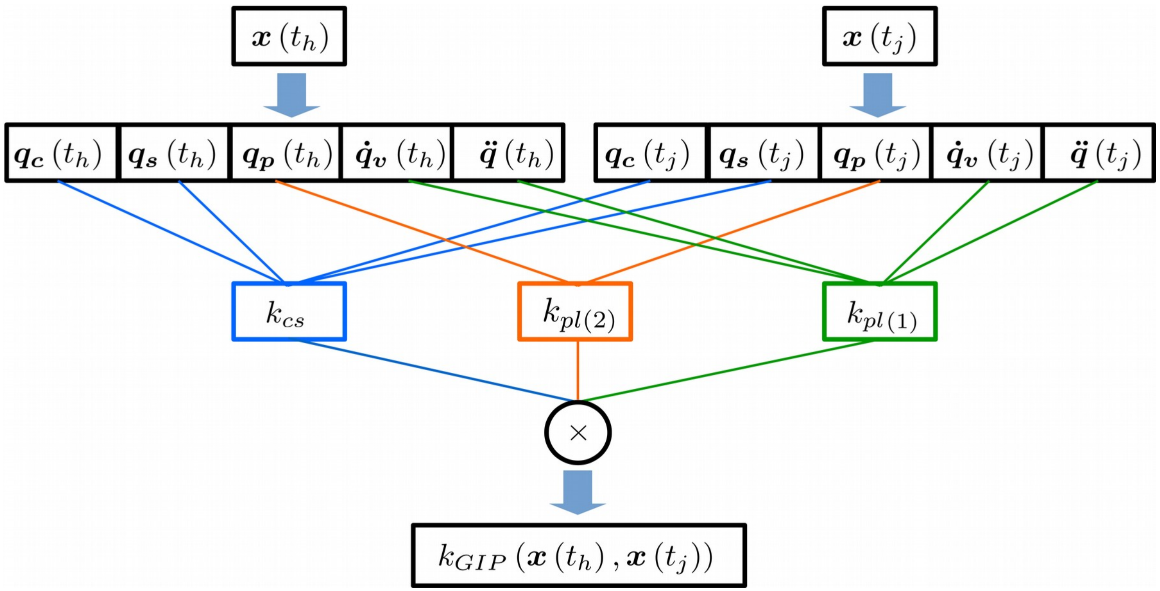
- The RKHS of the inhomogeneous Polynomia kernel is composed of all the monomials up to the polynomial degree
- The product of kernels is still a kernel, and its RKHS is given by the convolution of the RKHS of the two kernels

$$\hat{f} \text{ is the MAP estimator } \iff \hat{f} = \text{argmin}_{f \in \mathcal{H}} ||\boldsymbol{y}^{(k)} - \boldsymbol{f}(X)||^2 + \sigma_n^2 ||f||^2_{\mathcal{H}} \text{ with } \mathcal{H} \text{ RKHS of } k$$

$$k_{pl(p)}\left(\boldsymbol{a}(t_h), \boldsymbol{a}(t_j)\right) \rightarrow \mathbb{P}_{[p]}\left(\boldsymbol{a}_{[p]}\right)$$

$$k_{pl(p_a)}\left(\boldsymbol{a}(t_h), \boldsymbol{a}(t_j)\right) k_{pl(p_b)}\left(\boldsymbol{b}(t_h), \boldsymbol{b}(t_j)\right) \rightarrow \mathbb{P}_{[p_a+p_b]}\left(\boldsymbol{a}_{[p_a]}, \boldsymbol{b}_{[p_b]}\right)$$

GIP KERNEL



REFERENCE:

- IEEE Robotics and Automation Letters. PP. 1-1. 10.1109/LRA.2019.2945240.
  A. Dalla Libera, R. Carli. (2019). A Data-Efficient Geometrically Inspired Polynomial Kernel for Robot Inverse Dynamics.

GIP kernel: Derivation

KERNEL PROPERTIES (RKHS interpretation)

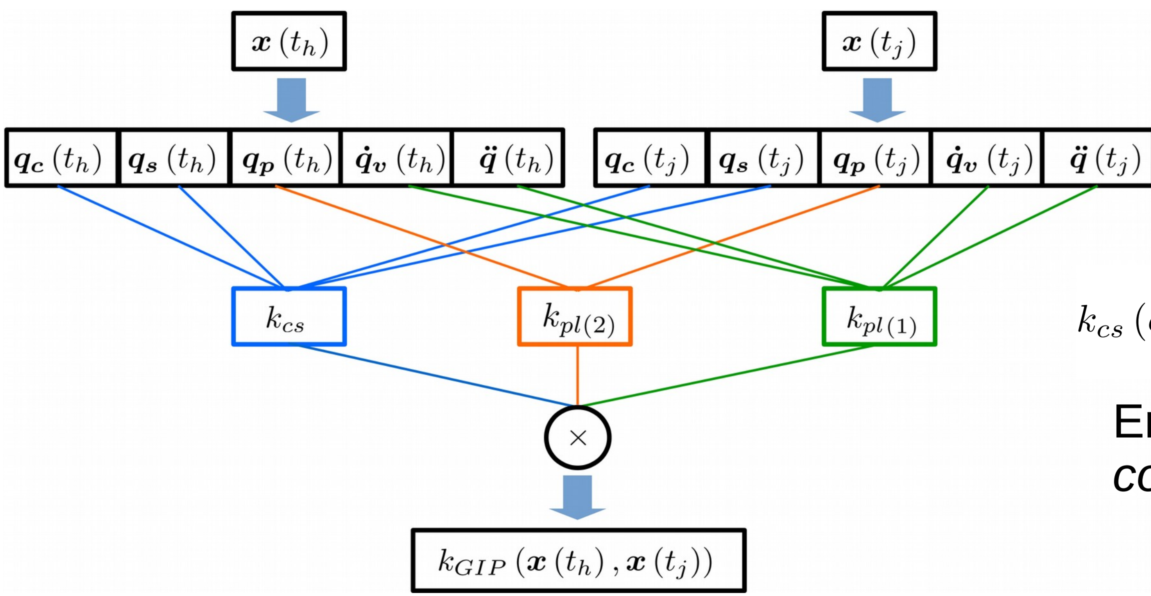
- The RKHS of the inhomogeneous Polynomia kernel is composed of all the monomials up to the polynomial degree
- The product of kernels is still a kernel, and its RKHS is given by the convolution of the RKHS of the two kernels

$$\hat{f} \text{ is the MAP estimator} \iff \hat{f} = \operatorname{argmin}_{f \in \mathcal{H}} ||\boldsymbol{y}^{(k)} - \boldsymbol{f}(X)||^2 + \sigma_n^2 ||f||_{\mathcal{H}}^2 \text{ with } \mathcal{H} \text{ RKHS of } k$$

$$k_{pl(p)}\left(\boldsymbol{a}(t_h), \boldsymbol{a}(t_j)\right) \to \mathbb{P}_{[p]}\left(\boldsymbol{a}_{[p]}\right)$$

$$k_{pl(p_a)}\left(\boldsymbol{a}(t_h), \boldsymbol{a}(t_j)\right) k_{pl(p_b)}\left(\boldsymbol{b}(t_h), \boldsymbol{b}(t_j)\right) \to \mathbb{P}_{[p_a+p_b]}\left(\boldsymbol{a}_{[p_a]}, \boldsymbol{b}_{[p_b]}\right)$$

GIP KERNEL



$$k_{cs}\left(\boldsymbol{q}_{cs}(t_h), \boldsymbol{q}_{cs}(t_j)\right) = \prod_{r=r_1}^{N_r} k_{pl(2)}\left(\begin{bmatrix} q_c^{(r)}(t_h) \\ q_s^{(r)}(t_h) \end{bmatrix}, \begin{bmatrix} q_c^{(r)}(t_j) \\ q_s^{(r)}(t_j) \end{bmatrix}\right)$$

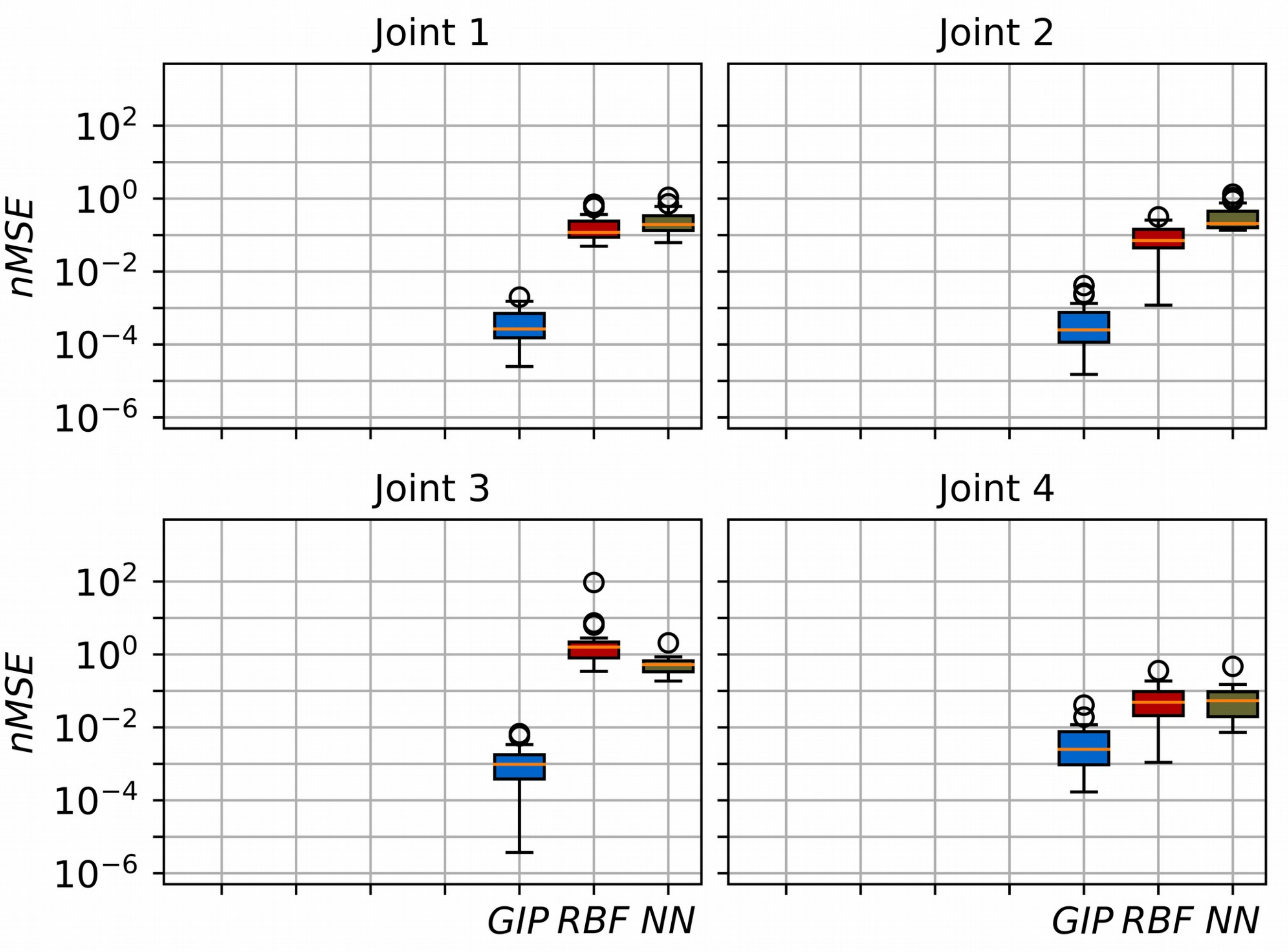
Encodes all the terms that depends on *cos* and *sin* satisfying:

$$deg\left(\boldsymbol{q}_c^{(i)}\right) + deg\left(\boldsymbol{q}_s^{(i)}\right) \leq 2$$

REFERENCE:

- IEEE Robotics and Automation Letters. PP. 1-1. 10.1109/LRA.2019.2945240.
  A. Dalla Libera, R. Carli. (2019). A Data-Efficient Geometrically Inspired Polynomial Kernel for Robot Inverse Dynamics.

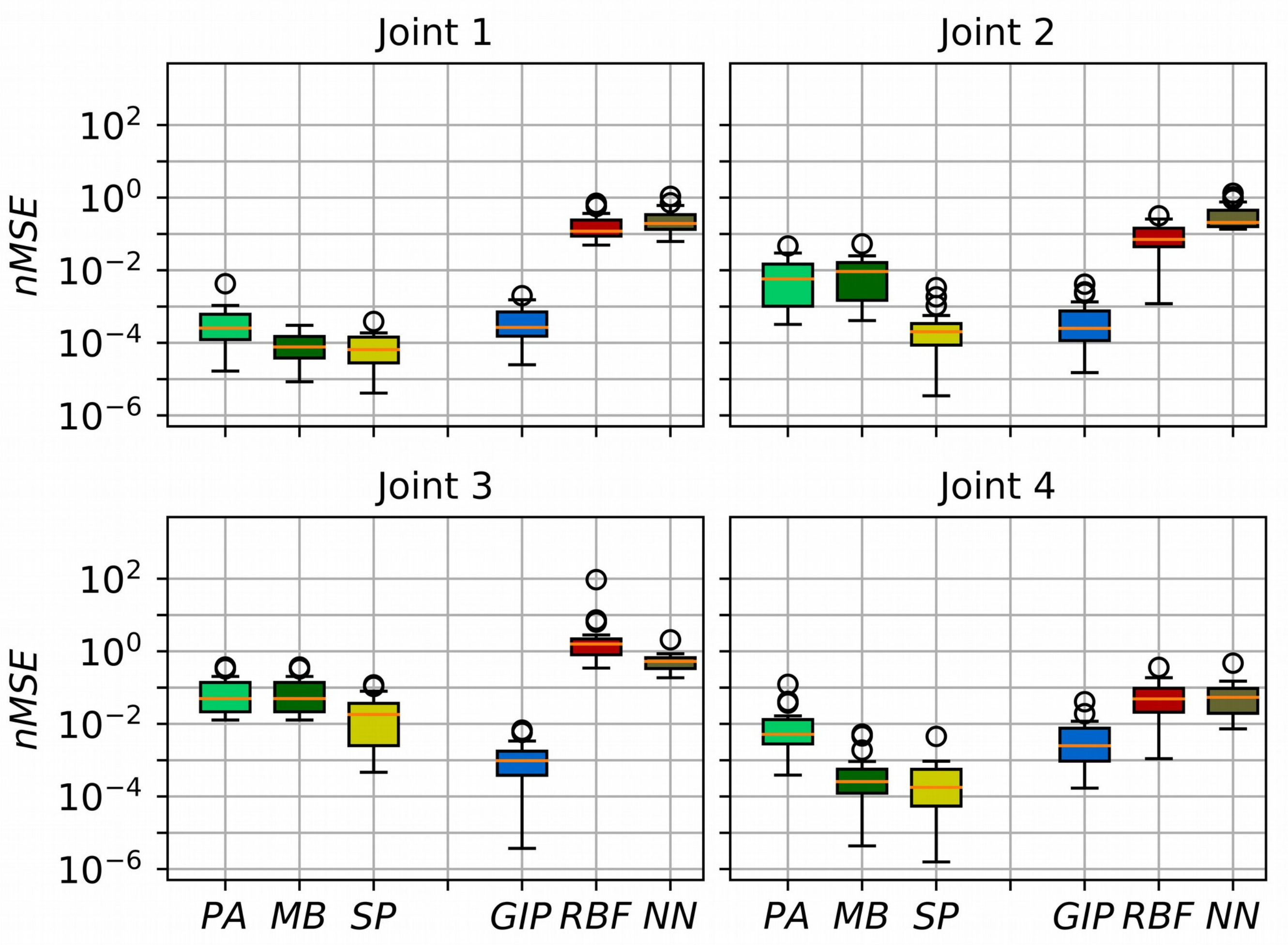
GIP kernel: Numerical results



MONTE CARLO EXPERIMENT:

- **Setup**: simulated SCARA robot
- **20 simulations**:
  - Training and test dataset: **2000 samples** (20 sec)
  - Joint trajectories: sum of **200 random sin**
- Measure of performance: **Normalized Mean Squared Error**
- Estimators compared:
  - **Model-free**:
    - GIP kernel
    - RBF kernel
    - NN: 2 layer neural network (400 sigmoids per layer)

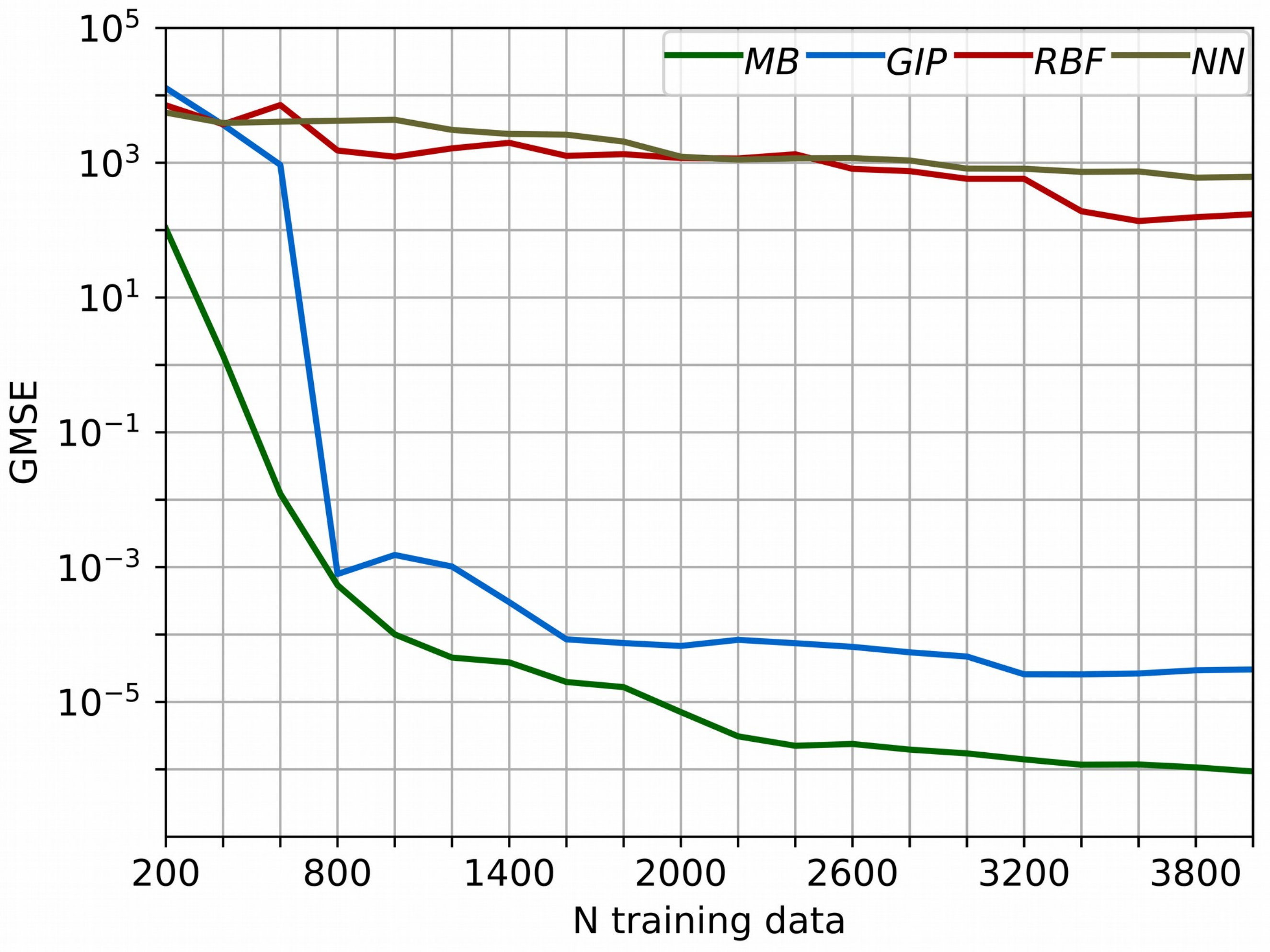
GIP kernel: Numerical results



MONTE CARLO EXPERIMENT:

- **Setup**: simulated SCARA robot
- **20 simulations**:
  - Training and test dataset: **2000 samples** (20 sec)
  - Joint trajectories: sum of **200 random sin**
- Measure of performance: **Normalized Mean Squared Error**
- Estimators compared:
  - **Model-free**:
    - GIP kernel
    - RBF kernel
    - NN: 2 layer neural network (400 sigmoids per layer)
  - **Model-Based** (with perturbation of the geometrical parameters):
    - PA: parametric approach
    - MB kernel
    - SP kernel

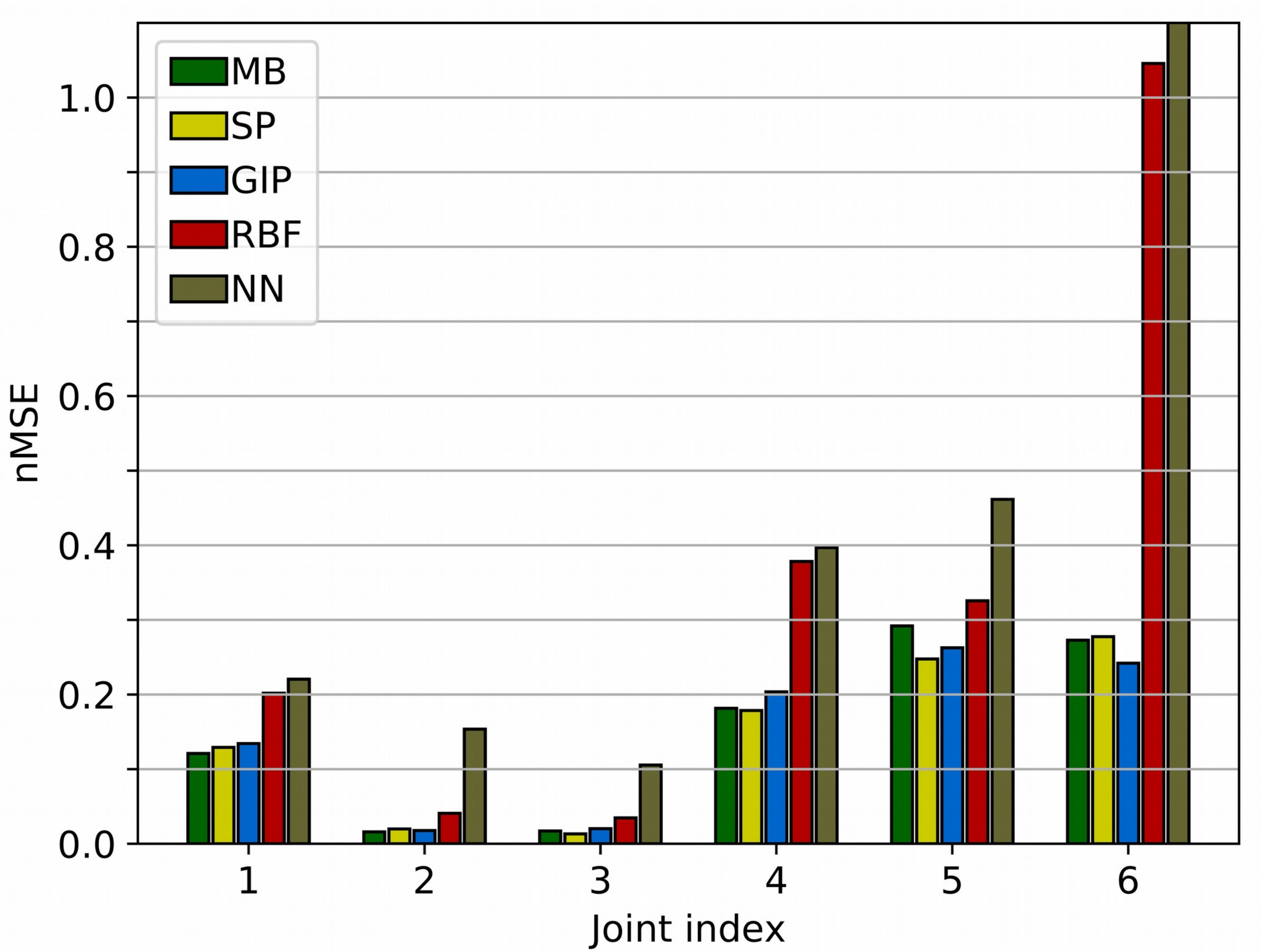
GIP kernel: Numerical results



DATE-EFFICIENCY TEST:

- **Setup**: simulated SCARA robot
- Measure of performance: **Global Mean Squared Error**
- Training and test dataset: 4000 samples (40 sec)
- Estimators compared:
  - **Model-free**:
    - GIP kernel
    - RBF kernel
    - NN: 2 layer neural network (400 sigmoids per layer)
  - **Model-Based** (without perturbation of the geometrical parameters):
    - MB kernel

GIP kernel: Numerical results



TEST WITH REAL DATA:

- **Setup**: UR10 robot
- Measure of performance: **Normalized Mean Squared Error**
- Training dataset: 40000 (random points)
- Test dataset: 25000 (random points+ circle)
- Estimators compared:
  - **Model-free**:
    - GIP kernel
    - RBF kernel
    - NN: 2 layer neural network (400 sigmoids per layer)
  - **Model-Based**:
    - MB kernel
    - SP kernel

Conclusion

- We have introduced different data-driven strategies which do not requires high prior knowledge about the robot model
- The problem considered are:
  - Kinematics (modeling and control)
  - Dynamics (proprioceptive contact detection)
  - RL-based control
- We Introduced the GIP kernel, a data-efficient kernel for inverse dynamics identification:
  - No need of strong prior information
  - Data-efficiency
  - Good generalization
  - Good asymptotic performance

THANKS FOR
THE ATTENTION