Optimal and Learning Control for Autonomous Robots

Lecture 12

A D R L

Jonas Buchli
Agile & Dexterous Robotics Lab

FNSNF

ETH Zürich

Class logistics

Exercise 3 hand out today!
Due: Tue. May 26 - 18.00

Interviews: Thu/Fri. May 28/29
https://ethz.doodle.com/bsi7gvkycvrmht6t

L11 Recap

Forward diffusion / sampling

Importance sampling

Function approximation

Linear Markov Decision Process

Three conditions on the optimal control problem:

1) Quadratic control cost

$$J = E\left\{\Phi(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} q(t,\mathbf{x}) + \frac{1}{2}\mathbf{u}^T\mathbf{R}\mathbf{u}\, dt\right\}$$

2) Control affine system

$$d\mathbf{x} = \mathbf{f}(t,\mathbf{x})dt + \mathbf{g}(t,\mathbf{x})\left(\mathbf{u}dt + d\mathbf{w}\right), \qquad d\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}dt)$$

$$\dot{\mathbf{x}} = \mathbf{f}(t,\mathbf{x}) + \mathbf{g}(t,\mathbf{x})\left(\mathbf{u} + \varepsilon\right), \qquad \varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$$

3) $\mathbf{R}\mathbf{\Sigma} = \lambda\mathbf{I}$

Linear Markov Decision Process (cnt)

nonlinear PDE

$$-\partial_t V^* = q - \frac{1}{2}\nabla_x^T V^* \,\Xi \nabla_x V^* + \nabla_x^T V^* \,\mathbf{f} + \frac{\lambda}{2}\mathtt{Tr}[\nabla_{xx} V^* \,\Xi]$$

$$V^*(t,\mathbf{x}) = -\lambda \log \Psi(t,\mathbf{x})$$

$$-\partial_t \Psi = -\frac{1}{\lambda} q\Psi + \mathbf{f}^T \nabla_x \Psi + \frac{\lambda}{2}\mathtt{Tr}[\Xi \nabla_{xx}\Psi]$$

$$-\partial_t \Psi = \mathtt{H}[\Psi] \qquad \mathtt{H} = -\frac{1}{\lambda} q + \mathbf{f}^T \nabla_x + \frac{\lambda}{2}\mathtt{Tr}[\Xi \nabla_{xx}]$$

$$\Psi(t_f,\mathbf{x}) = \exp\left(-\frac{1}{\lambda}\Phi(\mathbf{x})\right)$$

Final Value problem

$$\mathbf{g}\Sigma\mathbf{g}^T = \lambda \Xi$$

The effective Covariance

Path Integral

$$\Psi(s,\mathbf{y}) = \int \mathbb{P}_{uc}(\tau \mid s,\mathbf{y})\ \mathrm{e}^{-\frac{1}{\lambda}\left(\Phi(\mathbf{x}_N)+\sum_{i=0}^{N-1} q(t_i,\mathbf{x}(t_i))dt\right)}\ d\mathbf{x}(t_1)\dots d\mathbf{x}(t_{N-1})d\mathbf{x}_N$$

Equivalently

$$\Psi(s,\mathbf{y}) = \mathrm{E}_{\tau_{uc}}\left\{\mathrm{e}^{-\frac{1}{\lambda}\left(\Phi(\mathbf{x}(t_N))+\sum_{i=0}^{N-1} q(t_i,\mathbf{x}(t_i))dt\right)}\right\}$$

$$= \mathrm{E}_{\tau_{uc}}\left\{\mathrm{e}^{-\frac{1}{\lambda}\left(\Phi(\mathbf{x}(t_f))+\int_{t_0}^{t_f} q(t,\mathbf{x})\ dt\right)}\right\}$$

Samples can be generated by

$$d\mathbf{x} = \mathbf{f}(t,\mathbf{x})dt + \mathbf{g}(t,\mathbf{x})d\mathbf{w}, \qquad d\mathbf{w} \sim \mathcal{N}(\mathbf{0},\mathbf{\Sigma}dt), \quad \mathbf{x}(t=s) = \mathbf{y}$$

Closer look at Path Integral formula

- For calculating the Desirability function at each point

$$\Psi(s, \mathbf{y}) = \mathrm{E}_{\tau_{uc}} \left\{ \mathrm{e}^{-\frac{1}{\lambda}\left(\Phi(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} q(t,\mathbf{x})\ dt\right)} \right\}$$

$$d\mathbf{x} = \mathbf{f}(t, \mathbf{x})dt + \mathbf{g}(t, \mathbf{x})d\mathbf{w}, \qquad d\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma dt), \quad \mathbf{x}(t = s) = \mathbf{y}$$

1) Forward simulate the uncontrolled system from $(s, \mathbf{y})$ up to t_f
2) Integrate the cost over the generated path

Path Integral: Optimal Control

- Using the white noise formulation $\varepsilon = \frac{d\mathbf{w}}{dt}$

$$\dot{\mathbf{x}} = \mathbf{f}(t,\mathbf{x}) + \mathbf{g}(t,\mathbf{x})\varepsilon, \qquad \varepsilon \sim \mathcal{N}(\mathbf{0},\boldsymbol{\Sigma}), \quad \mathbf{x}(t=s) = \mathbf{y}$$

$$\mathbf{u}^*(s,\mathbf{y}) = \frac{\mathrm{E}_{\tau_{uc}}\left\{\varepsilon \;\; \mathrm{e}^{-\frac{1}{\lambda}\left(\Phi(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} q(t,\mathbf{x})\ dt\right)}\right\}}{\mathrm{E}_{\tau_{uc}}\left\{\mathrm{e}^{-\frac{1}{\lambda}\left(\Phi(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} q(t,\mathbf{x})\ dt\right)}\right\}}$$

A D R L

ETH Zürich

Example: Naive sampling

$$M(\theta)\cdot\ddot{\theta}+C(\theta,\dot{\theta})=\tau$$

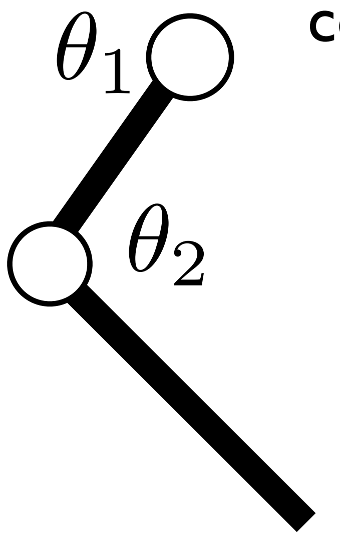
$$\ddot{\theta}=M(\theta)^{-1}\cdot\left(-C(\theta,\dot{\theta})+\tau\right)$$

$$M(\theta) = \begin{pmatrix} d_1+2d_2\cos(\theta_2) & d_3+d_2\cos(\theta_2) \\ d_3+d_2\cos(\theta_2) & d_3 \end{pmatrix} \quad (37)$$

$$C(\dot{\theta},\theta) = \begin{pmatrix} -\dot{\theta}_2\left(2\dot{\theta}_1+\dot{\theta}_2\right) \\ \dot{\theta}_1^2 \end{pmatrix} d_2\sin(\theta_2) \quad (38)$$

$$d_1 = I_1+I_2+m_2 l_1^2, \quad d_2 = m_2 l_1 s_2, \quad d_3 = I_2 \quad (39)$$

Symbol	Value	Unit
m_1	1.4	Kg
m_2	1	Kg
s_1	0.11	m
s_2	0.16	m
I_1	0.3	Kg m^2
I_2	0.33	Kg m^2
l_1	0.025	m
l_2	0.045	m

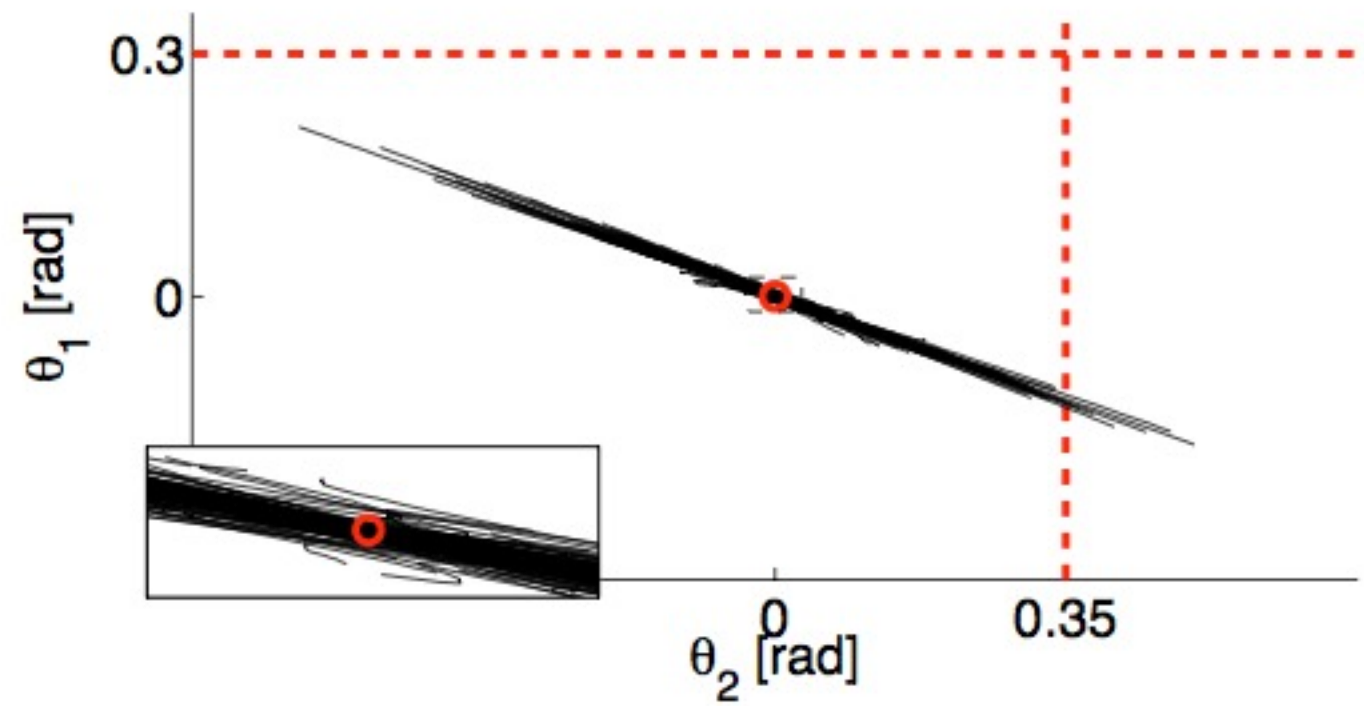


Compare to affine control assumption

$$\dot{x}=\Phi(x)+G(x)\cdot\tau$$

$$x = \left(\theta_1\ \theta_2\ \dot{\theta}_1\ \dot{\theta}_2\right)^T \quad \tau = \left(\tau_1\ \tau_2\right).$$

$$\Phi(x) = \begin{pmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ -M(\theta)^{-1}\cdot C(\theta,\dot{\theta}) \end{pmatrix} \quad G(x) = \begin{pmatrix} O_{2\times 2} \\ M(\theta)^{-1} \end{pmatrix}$$



Path integral SOC requires sampling of passive dynamics with gaussian mean-free noise

A D R L

Bucl

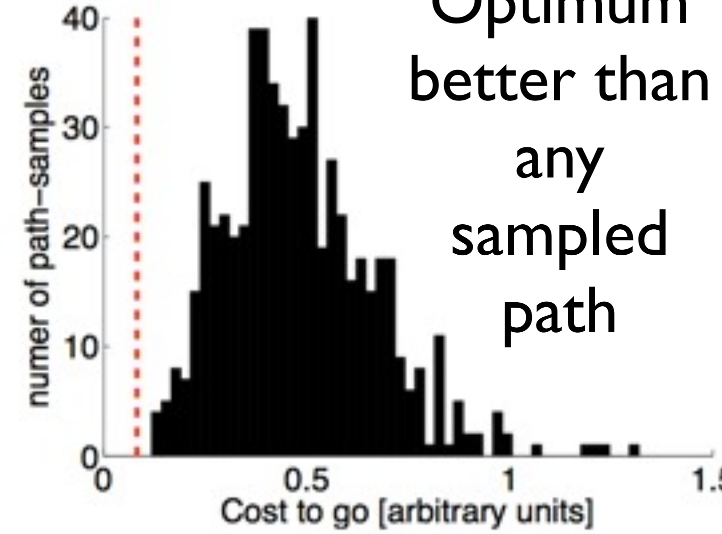
Improved sampling...?

i.d. - controller

$$\tau_u^i = M(\theta) \cdot (\alpha_i + \epsilon_i) + C(\theta, \dot{\theta})$$

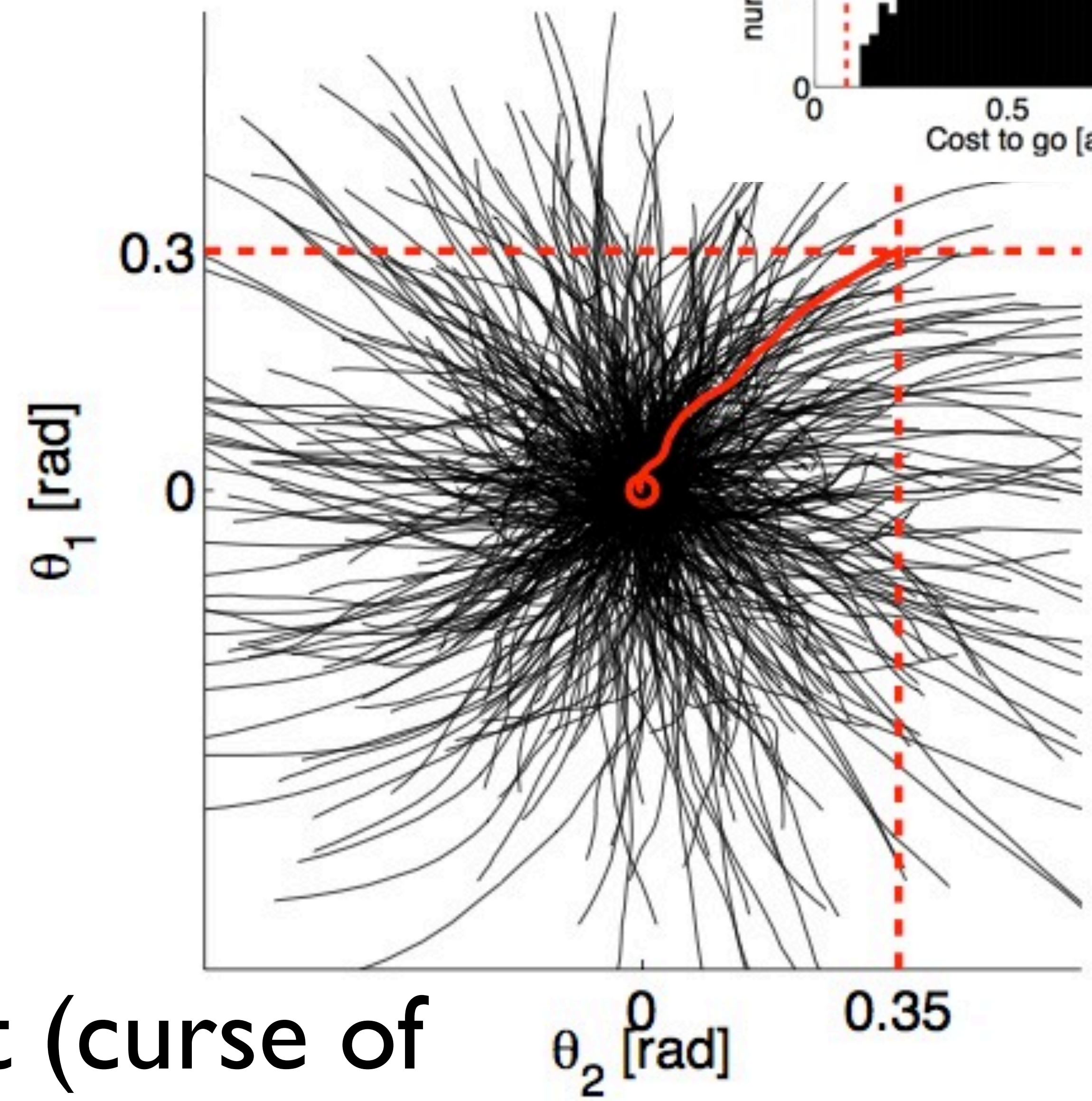
$$M(\theta) \cdot \ddot{\theta} + C(\theta, \dot{\theta}) = \tau_u$$

plant dynamics



Optimum better than any sampled path

Sample in acceleration space, use inverse dynamics controllers to find torques:



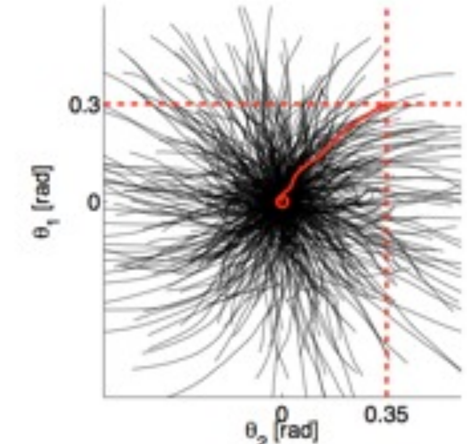
... still not very efficient (curse of dimensionality still strikes, needle in a haystack!)

A D R L

ETH Zürich

Path Integral: issues (1)

- Inefficient sampling



Soft Max

$$\mathbf{u}^*(s,\mathbf{y}) = \mathrm{E}_{\tau_{uc}}\left\{ \varepsilon \frac{e^{-\frac{1}{\lambda}\left(\Phi(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} q(t,\mathbf{x})\ dt\right)}}{\mathrm{E}_{\tau_{uc}}\left\{ e^{-\frac{1}{\lambda}\left(\Phi(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} q(t,\mathbf{x})\ dt\right)} \right\}} \right\}$$

It just has significant value for near optimal solution

What are the chances to hit the optimal solution by a random walk?

Importance Sampling

Path Integral: issues (2)

- Point-wise estimation of the optimal controls

$$\mathbf{u}^*(s,\mathbf{y}) = \mathrm{E}_{\tau_{uc}}\left\{ \varepsilon \frac{\mathrm{e}^{-\frac{1}{\lambda}\left(\Phi(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} q(t,\mathbf{x})\ dt\right)}}{\mathrm{E}_{\tau_{uc}}\left\{\mathrm{e}^{-\frac{1}{\lambda}\left(\Phi(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} q(t,\mathbf{x})\ dt\right)}\right\}} \right\}$$

The optimal control is estimated independently for each point

Does the optimal control change drastically from one point to the other?

Function Approximation

Reward weighting

$$\mathbf{u}^*(s,\mathbf{y}) = \mathrm{E}_{\tau_{uc}}\left\{\varepsilon \frac{\mathrm{e}^{-\frac{1}{\lambda}\left(\Phi(\mathbf{x}(t_f))+\int_{t_0}^{t_f} q(t,\mathbf{x})\ dt\right)}}{\mathrm{E}_{\tau_{uc}}\left\{\mathrm{e}^{-\frac{1}{\lambda}\left(\Phi(\mathbf{x}(t_f))+\int_{t_0}^{t_f} q(t,\mathbf{x})\ dt\right)}\right\}}\right\}$$

$$\alpha(\tau_{uc}; s, \mathbf{y}) = \frac{\mathrm{e}^{-\frac{1}{\lambda}\left(\Phi(\mathbf{x}(t_f))+\int_{t_0}^{t_f} q(t,\mathbf{x})\ dt\right)}}{\mathrm{E}_{\tau_{uc}}\left\{\mathrm{e}^{-\frac{1}{\lambda}\left(\Phi(\mathbf{x}(t_f))+\int_{t_0}^{t_f} q(t,\mathbf{x})\ dt\right)}\right\}}$$

$$\boldsymbol{\theta}_i^* = \boldsymbol{\theta}_{i,c} + \underset{\Delta\boldsymbol{\theta}_i}{\mathrm{argmin}} \int \frac{\mathrm{e}^{-\frac{1}{\lambda}R(\tau;s,\mathbf{y})}}{\mathrm{E}_{\tau_c}\left\{\mathrm{e}^{-\frac{1}{\lambda}R(\tau;s,\mathbf{y})}\right\}} \|\boldsymbol{\Upsilon}_i^T(s,\mathbf{y})\Delta\boldsymbol{\theta}_i - \varepsilon\|_2^2\ \mathbb{P}_{\tau_c}(\tau \mid s,\mathbf{y}) p(s,\mathbf{y}) d\tau d\mathbf{y} ds$$

A D R L

ETH Zürich

EOF Recap

L12

Lecture 12 Goals

★Policy Improvement with Path Integrals - PI2
★Combination of optimal and learning control

General Path Integral Algorithm

Importance Sampling

Goal: Sample more efficiently

Update sampling distribution to account what is already known about good solutions

Use current estimate of optimal controls for sampling

Function approximation

Goal: Reduce complexity by generalizing

Function approximation reduces open parameters

Each parameter covers a neighborhood

Algorithm 8 General Path Integral Algorithm

given

The cost function:

$J = \Phi(\mathbf{x}(t_f)) + \int_s^{t_f} \left(q(t,\mathbf{x}) + \frac{1}{2}\mathbf{u}^T\mathbf{R}\mathbf{u}\right) dt$

A PDF defining the quality of approximation of optimal control at each time-state pair: $p(t,\mathbf{x})$

An initial policy and a Linear Model: $\mathbf{u}(t,\mathbf{x}) = [u_i(t,\mathbf{x})] = [\boldsymbol{\Upsilon}_i^T(t,\mathbf{x})\boldsymbol{\theta}_i]$

function approximation

repeat

(a) Randomly choose a time-state pair from $p(t,\mathbf{x})$: $(s,\mathbf{y})$

(b) Forward simulate the controlled system for K different rollouts: $\{\tau^k\}_{k=1}^K$

sampling repeatedly

$\dot{\mathbf{x}} = \mathbf{f}(t,\mathbf{x}) + \mathbf{g}(t,\mathbf{x})\,(\mathbf{u}+\boldsymbol{\varepsilon})$

$\mathbf{u}(t,\mathbf{x}) = [\boldsymbol{\Upsilon}_i^T(t,\mathbf{x})\boldsymbol{\theta}_i],\ \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0},\boldsymbol{\Sigma}),\ \mathbf{x}(t=s) = \mathbf{y}$

using current estimate of opt. control to sample

(c) Calculate the return for each rollout: $\{R^k\}_{k=1}^K$

$R(\tau;s,\mathbf{y}) = \Phi(\mathbf{x}(t_f)) + \int_s^{t_f} \left(q(t,\mathbf{x}) + \frac{1}{2}\mathbf{u}^T\mathbf{R}\mathbf{u}\right) dt + \int_s^{t_f} \mathbf{u}^T\mathbf{R}d\mathbf{w}$

(d) Calculate $\{\alpha^k\}_{k=1}^K$

$\alpha^k(s,\mathbf{y}) = \exp(-\frac{1}{\lambda}R^k) / \frac{1}{K}\sum_{j=1}^{K}\exp(-\frac{1}{\lambda}R^j)$

weighting by reward

reward weighted regression

(e) Solve the following linear regression problem for each control input i:

$\Delta\boldsymbol{\theta}_i = \operatorname{argmin} \sum_{k=1}^{K} \alpha^k \|\boldsymbol{\Upsilon}_i^T(s,\mathbf{y})\Delta\boldsymbol{\theta}_i - \varepsilon_i^k(s)\|_2^2$

(f) Update the parameter vector for each control input i:

$\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i + \omega\Delta\boldsymbol{\theta}_i$

until convergence

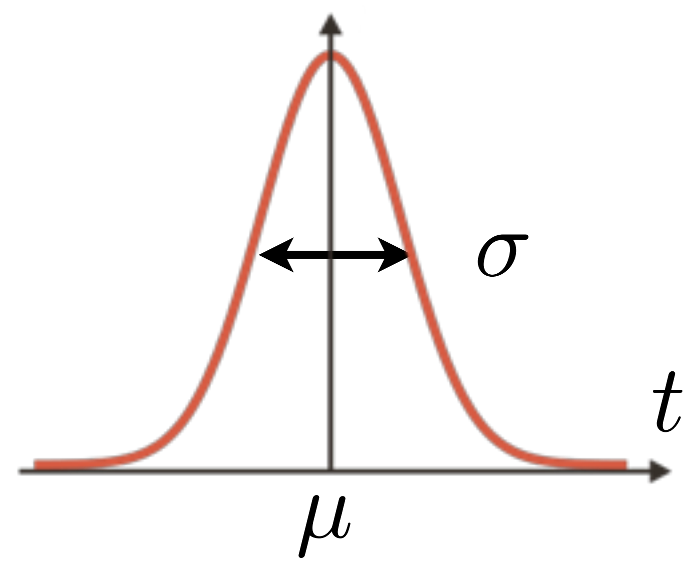
Policy improvement with Path Integrals - PI^2

In contrast ot the General Path Integral Algorithm, PI2 assumes that samples are extracted over the entire time horizon, not only for a single time step. Therefore the regression problem should be over different time steps as well as the rollout batch.

PI² w. time dependent policy

$$u_i(t) = \boldsymbol{\Upsilon}^T(t)\boldsymbol{\theta}_i$$

$$\boldsymbol{\Upsilon}(t) = [\Upsilon_n(t)]_{N\times 1} = \left[e^{-\frac{1}{2}\frac{(t-\mu_n)^2}{\sigma_n^2}} \right]_{N\times 1}$$



PI^2 Assumptions

1. The weighting function of MSE, $p(t, \mathbf{x})$, is assumed to be the probability distribution of the state under the latest estimation of the optimal control. Therefore the rollout trajectory's states are considered to be extracted form $p(t, \mathbf{x})$.

2. The return function, $R(\tau; s, \mathbf{y})$, won't be a function of state, if the system has been initialized from a similar initial condition. An immediate result of this assumption is that $\alpha^k(s, \mathbf{y})$ is only a function of time i.e. $\alpha^k(s)$. Therefore for a batch of trajectories extracted form a similar initial condition, $\alpha^k(s)$ can be estimated for each time step by the following

$$\alpha^k(s) = \frac{\exp(-\frac{1}{\lambda} R^k(s))}{\frac{1}{K}\sum_{j=1}^{K} \exp(-\frac{1}{\lambda} R^j(s))} \tag{3.60}$$

PI^2 Assumptions (cont'd)

3. The basis function vector for all the control inputs is the same. Hence we will drop the i subscription of the basis function as $[u_i(t)] = [\boldsymbol{\Upsilon}^T(t)\boldsymbol{\theta}_i]$.

4. Instead of adding noise to the control input, the noise is added directly to the parameter vector. Therefore the input noise will be as $\varepsilon_i = \boldsymbol{\Upsilon}(t)\boldsymbol{\epsilon}_i$, where $\boldsymbol{\epsilon}_i$ is the noise that is added to the parameter vector of the ith control input.

5. The PI2 regression problem should be modified as follows

$$\Delta\boldsymbol{\theta}_i = \mathrm{argmin} \sum_{s=t_0}^{t_f} \sum_{k=1}^{K} \alpha^k \|\boldsymbol{\Upsilon}_i^T(s)\Delta\boldsymbol{\theta}_i - \varepsilon_i^k(s)\|_2^2$$

In contrast ot the General Path Integral Algorithm, PI2 assumes that samples are extracted over the entire time horizon, not only for a single time step. Therefore the regression problem should be over different time steps as well as the rollout batch.

PI² - Regression step

In order to solve this regression problem, PI2 breaks it into two separate optimizations. This method finds the optimal solution as long as we can assume the regression error has a zero mean over the samples. In this method first the optimization is solved for each time step separately. Therefore the first optimization will find a time-dependent parameter vector increment that has the minimum error over the rollouts at each time step. Finally in the second optimization, we will find a parameter vector increment that approximates the time-dependent one.

1) min. error at each time step

The first optimization is defined as follows for each time step s

$$\Delta\boldsymbol{\theta}_i^*(s) = \operatorname{argmin} \sum_{k=1}^{K} \alpha^k(s) \| \boldsymbol{\Upsilon}_i^T(s) \Delta\boldsymbol{\theta}_i - \varepsilon_i^k(s) \|_2^2$$

PI^2 - Regression step (cont'd)

The first optimization is defined as follows for each time step s

$$\Delta\boldsymbol{\theta}_i^*(s) = \operatorname{argmin} \sum_{k=1}^{K} \alpha^k(s) \| \boldsymbol{\Upsilon}_i^T(s)\Delta\boldsymbol{\theta}_i - \varepsilon_i^k(s)\|_2^2$$

$$\Delta\boldsymbol{\theta}_i^*(s) = \sum_{k=1}^{K} \alpha^k(s) \frac{\boldsymbol{\Upsilon}_i(s)}{\boldsymbol{\Upsilon}_i^T(s)\boldsymbol{\Upsilon}_i(s)} \varepsilon_i^k(s)$$

$$\Delta\boldsymbol{\theta}_i^*(s) = \sum_{k=1}^{K} \alpha^k(s) \frac{\boldsymbol{\Upsilon}_i(s)\boldsymbol{\Upsilon}_i^T(s)}{\boldsymbol{\Upsilon}_i^T(s)\boldsymbol{\Upsilon}_i(s)} \boldsymbol{\epsilon}_i^k(s)$$

PI^2 - Regression step (cont'd)

2) parameter vect. increment approximating time dependent update

The second optimization for finding the optimal $\Delta\boldsymbol{\theta}_i^*$ is defined in equation (3.65). The index n refers to the nth element of the vector $\Delta\boldsymbol{\theta}_i^* = [\Delta\theta_{i,n}^*]$

$$\Delta\theta_{i,n}^* = \underset{\Delta\theta_{i,n}}{\operatorname{argmin}} \sum_{s=t_0}^{t_f} (\Delta\theta_{i,n} - \Delta\theta_{i,n}^*(s))^2 \Upsilon_n(s) \tag{3.65}$$

$\Upsilon_n(t)$ is the nth element of the basis function vector $\boldsymbol{\Upsilon}(t)$

$$\Delta\theta_{in}^* = \frac{\sum_{t=t_0}^{t_f} \Delta\theta_{in}^*(s)\Upsilon_n(s)}{\sum_{t=t_0}^{t_f} \Upsilon_n(s)}$$

A D R L

ETH Zürich

$$\Delta\theta^*_{in} = \frac{\sum_{t=t_0}^{t_f} \Delta\theta^*_{in}(s)\Upsilon_n(s)}{\sum_{t=t_0}^{t_f} \Upsilon_n(s)}$$

Compact notation: Use element-wise multiplication replace sum w. integral

$$\boldsymbol{\Delta\theta}^*_i = \left(\int_{t_0}^{t_f} \boldsymbol{\Delta\theta}^*_i(s) \circ \boldsymbol{\Upsilon}(s)ds\right) \cdot / \int_{t_0}^{t_f} \boldsymbol{\Upsilon}(s)ds$$

where $\circ$ and $\cdot/$ are element-wise multiplication and division.

Algorithm 9 PI2 Algorithm for time-dependent policy

given

The cost function:

$$J = \Phi(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} \left(q(t, \mathbf{x}) + \frac{1}{2}\mathbf{u}^T\mathbf{R}\mathbf{u}\right) dt$$

A Linear Model for function approximation: $\mathbf{u}(t) = [u_i(t)] = [\boldsymbol{\Upsilon}^T(t)\boldsymbol{\theta}_i]$

Initialize $[\boldsymbol{\theta}_i]$ with a sophisticated guess

Initialize exploration noise standard deviation: c

repeat

Create K rollouts of the system with the perturbated parameter $[\boldsymbol{\theta}_i] + [\boldsymbol{\epsilon}_i]$, $\quad \boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, c^2\mathbf{I})$

for the ith control input **do**

for each time, s **do**

Calculate the Return from starting time s for the kth rollout:

$$R(\tau^k(s)) = \Phi(\mathbf{x}(t_f)) + \int_s^{t_f} \left(q(t, \mathbf{x}) + \frac{1}{2}\mathbf{u}^T\mathbf{R}\mathbf{u}\right) dt$$

Calculate α from starting time s for the kth rollout:

$$\alpha^k(s) = \exp\left(-\frac{1}{\lambda}R(\tau^k(s))\right) / \sum_{k=1}^{K} \exp\left(-\frac{1}{\lambda}R(\tau^k(s))\right)$$

time dependent parameter increment

Calculate the time varying parameter increment $\Delta\boldsymbol{\theta}_i(s)$:

$$\Delta\boldsymbol{\theta}_i(s) = \sum_{k=1}^{K} \alpha^k(s) \frac{\boldsymbol{\Upsilon}(s)\boldsymbol{\Upsilon}^T(s)}{\boldsymbol{\Upsilon}^T(s)\boldsymbol{\Upsilon}(s)} \boldsymbol{\epsilon}_i^k(s)$$

end for

Time-averaging the parameter vector

$$\Delta\boldsymbol{\theta}_i = \left(\int_{t_0}^{t_f} \Delta\boldsymbol{\theta}_i(s) \circ \boldsymbol{\Upsilon}(s) ds\right) \cdot \Big/ \int_{t_0}^{t_f} \boldsymbol{\Upsilon}(s) ds$$

time averaging of increment

Update parameter vector for control input i, $\boldsymbol{\theta}_i$:

$$\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i + \omega\Delta\boldsymbol{\theta}_i$$

end for

- Decrease c for noise annealing

until maximum number of iterations

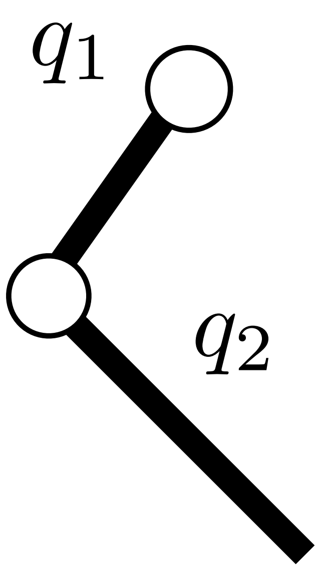
A D R L

ETH Zürich

Simple example

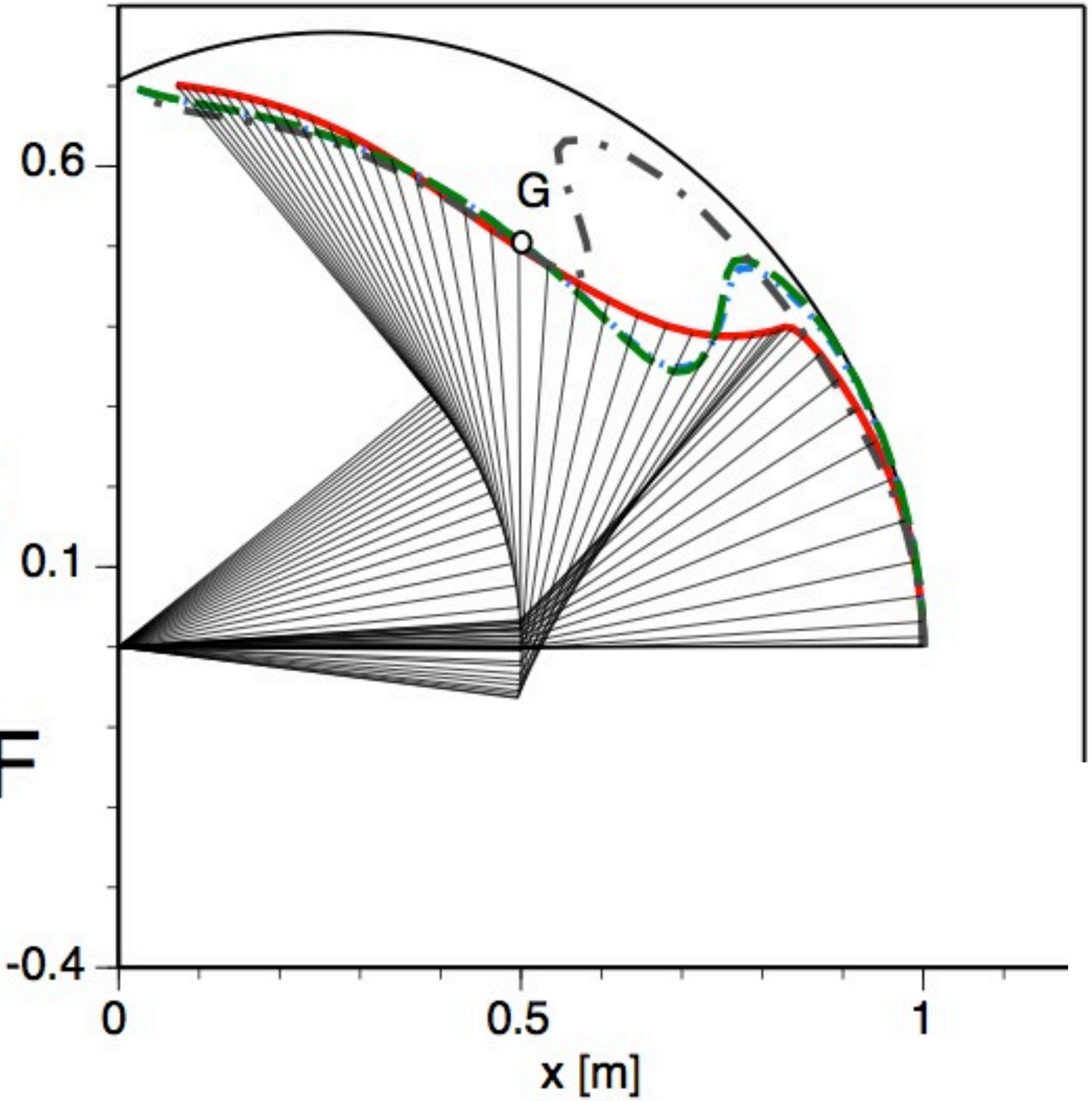
Assume position PID controller in each joint

$$\tau_i = -K_{p,i}(q_i - q_{i,des}) - K_{d,i}(\dot{q}_i - \dot{q}_{i,des})$$



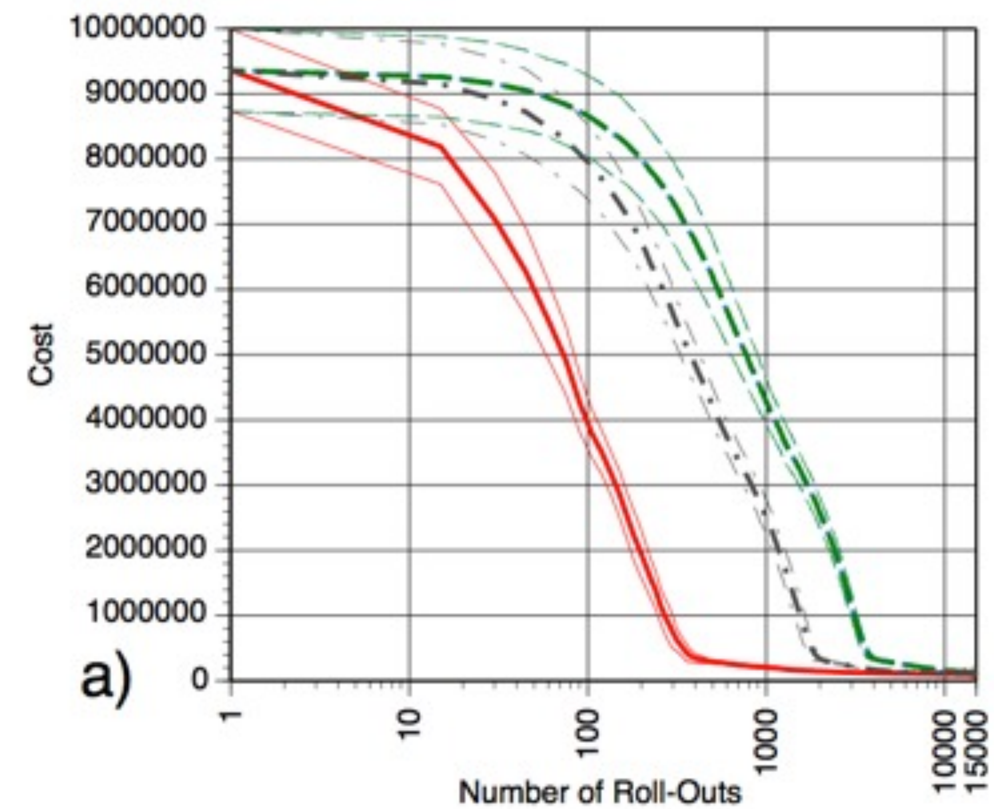
Parametrize learning problem via reference trajectories

$$q_{des,i}(t) = \boldsymbol{\Upsilon}(t)^T \boldsymbol{\theta}_{q,i}$$


0.6
G
y [m]
0.1
-0.4
0
0.5
1
x [m]

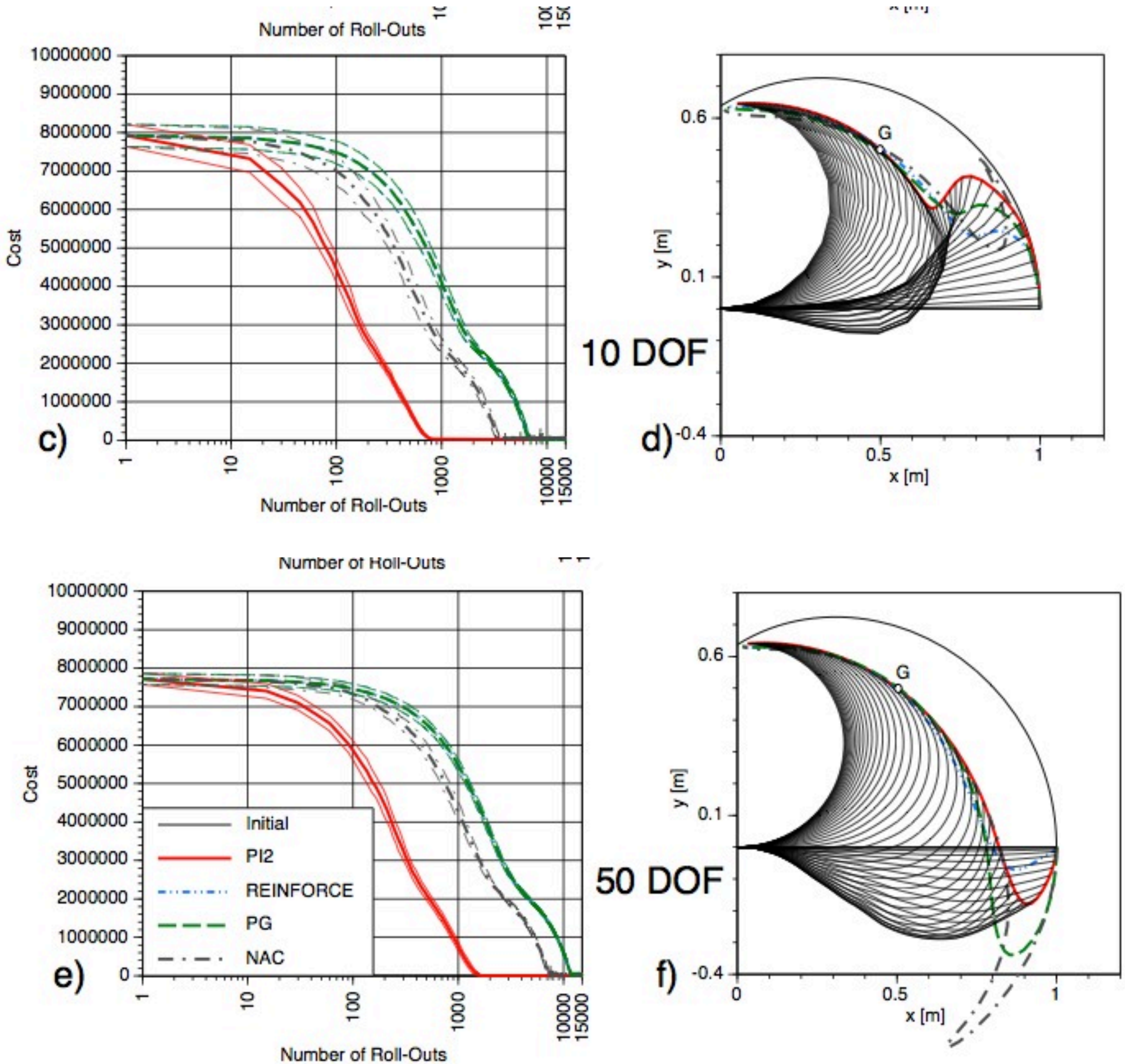
2 DOF

b)


10000000
9000000
8000000
7000000
6000000
5000000
4000000
3000000
2000000
1000000
0
Cost
1
10
100
1000
10000
15000
Number of Roll-Outs

a)

[Theodorou et al, 2010]



Learning Feedback Gains with PI2

Consider system with tracking controller

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}) + \mathbf{g}(t, \mathbf{x})\mathbf{u} \\ \mathbf{u} = \mathbf{K}^T(t)(\mathbf{x} - \mathbf{x}_{ref}) \end{cases}$$

$$\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}) + \mathbf{g}(t, \mathbf{x})(\mathbf{x} - \mathbf{x}_{ref})^T \mathbf{K}(t)$$

Treat gains K as controls - use PI2 for time indexed policies

Learning variable impedance control

Application of gain learning to rigid body dynamics / Torque controlled robots

x

Position control

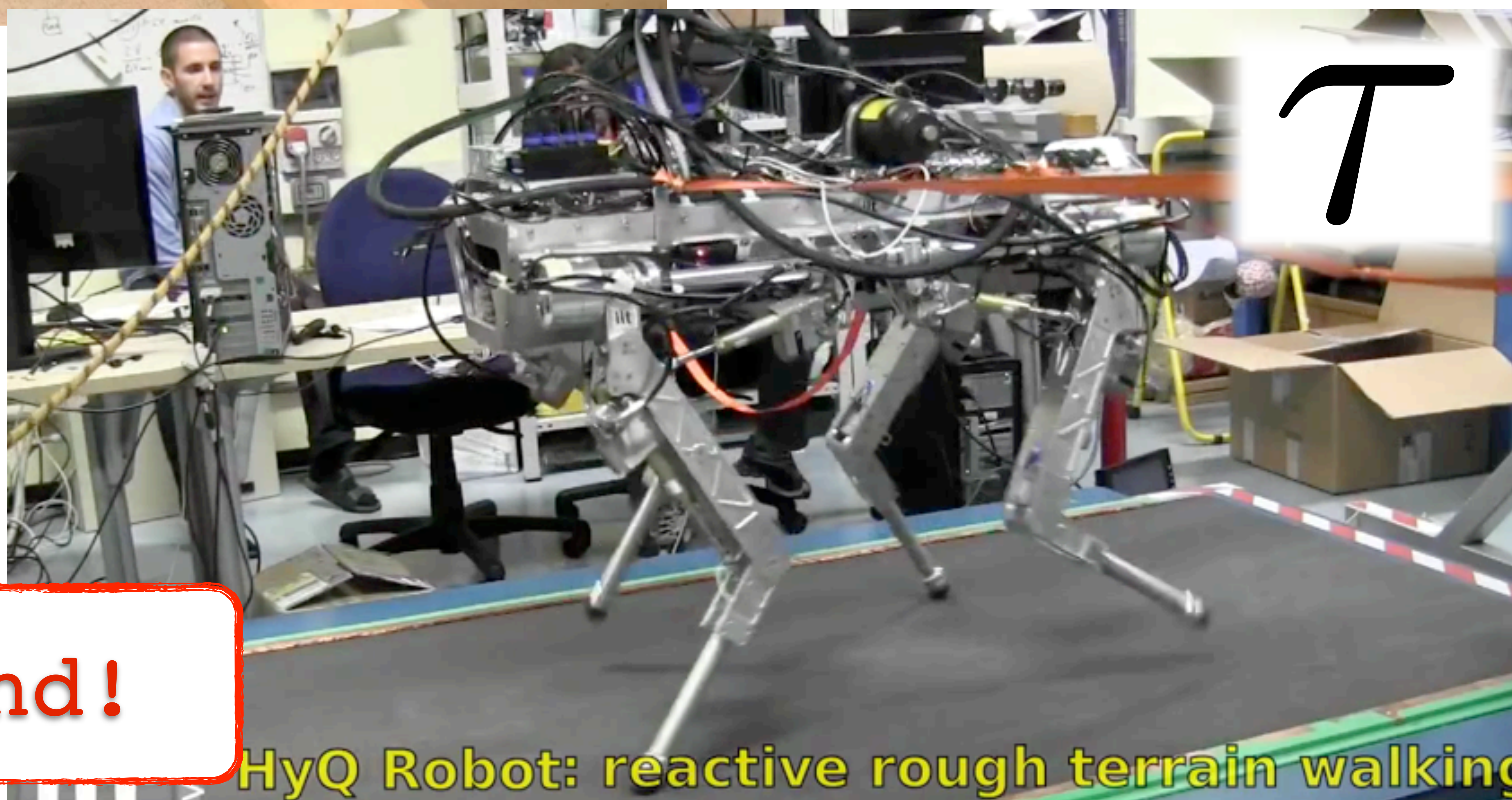
only kinematic model

high gains / 'stiff'

low gains / 'soft'

Torque control

+ dynamics model



τ

Blind!

Why?

Consider a robot with arms and legs:

- Low gains help to make the robot more robust in unknown surroundings
- Reduce energy consumption
- But position tracking is reduced

Thus, try to chose appropriate gain for the given situation. The requirements changes over time. Need a gain schedule.

Model of a simple robot

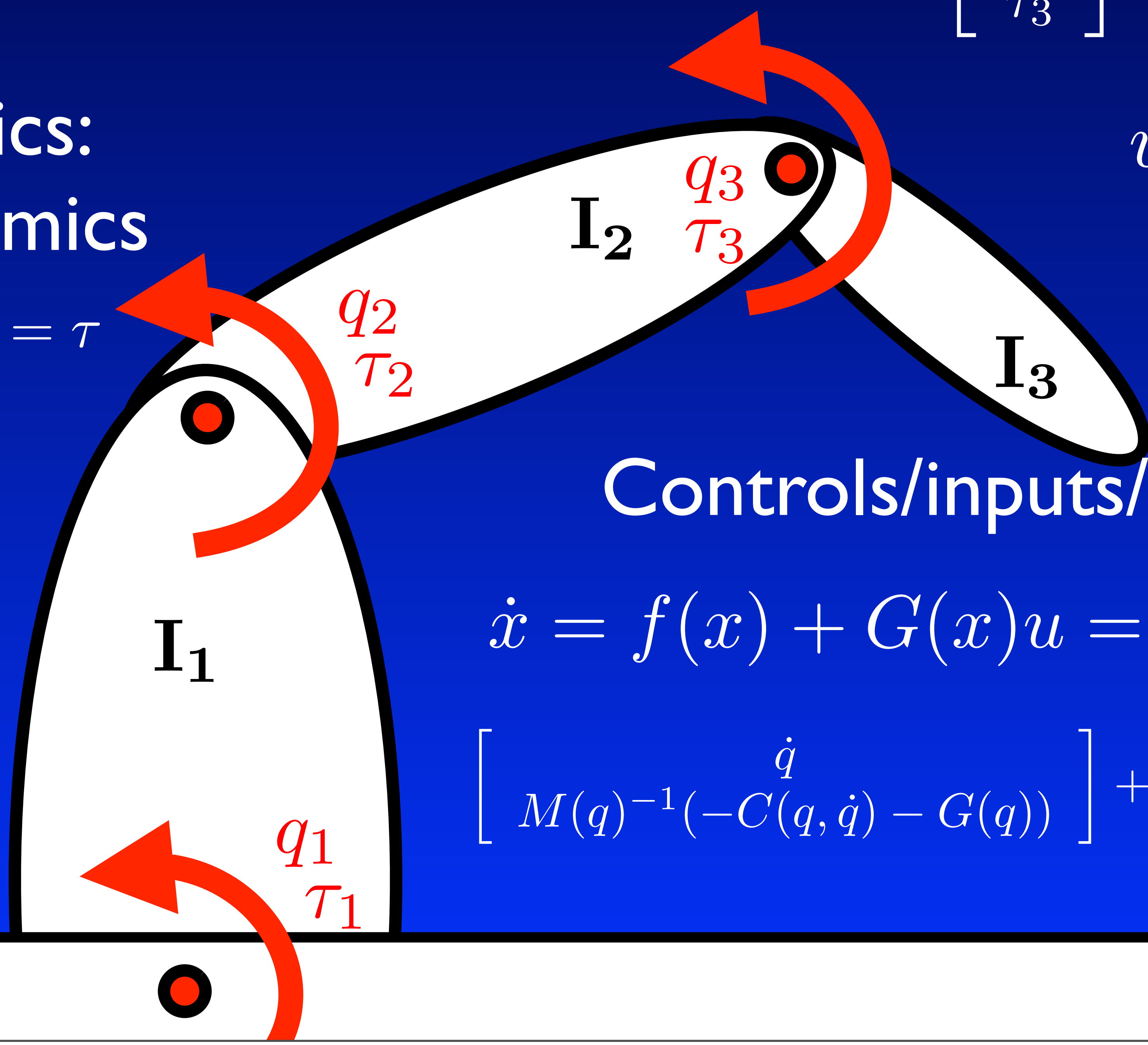
Measurements: Joint angles: $q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix}$

Actuation: joint torques $\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix}$

$u = \tau$

Governing physics:
Rigid body dynamics

$$M\ddot{q} + C(q,\dot{q}) + G(q) = \tau$$



System states:

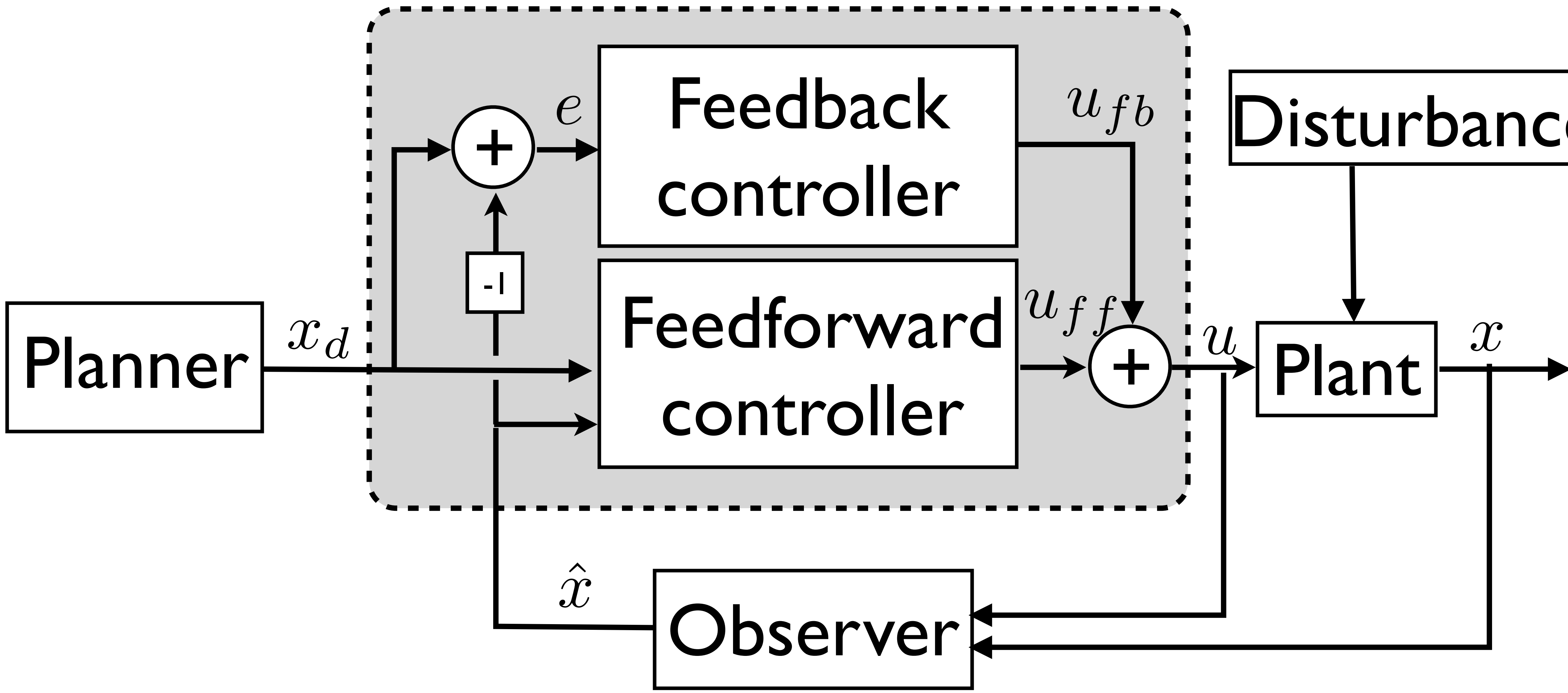
$$x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}$$

Controls/inputs/actions:

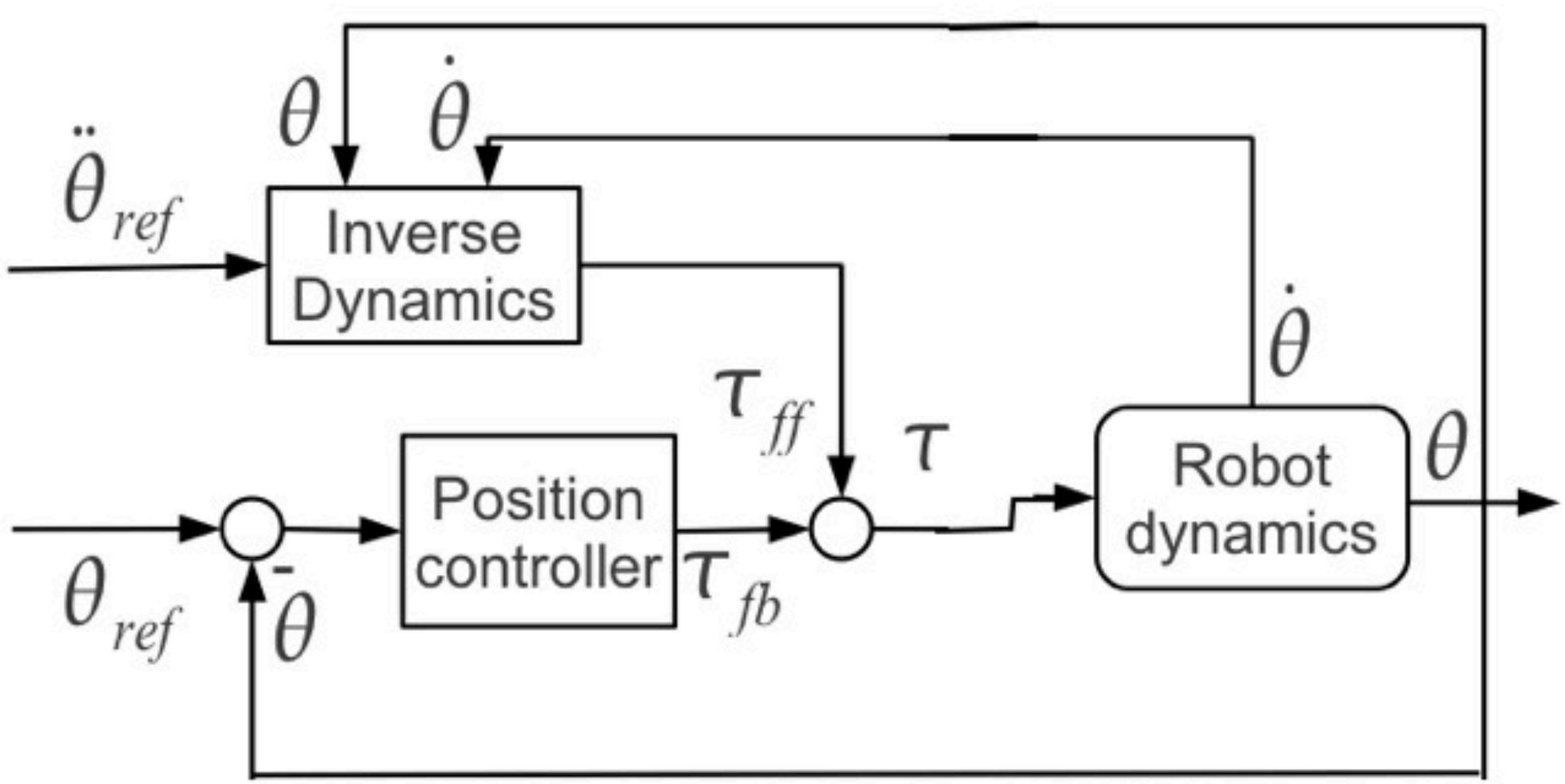
$$\dot{x} = f(x) + G(x)u =$$

$$\begin{bmatrix} \dot{q} \\ M(q)^{-1}(-C(q,\dot{q}) - G(q)) \end{bmatrix} + \begin{bmatrix} 0 \\ M(q)^{-1} \end{bmatrix}^T u$$

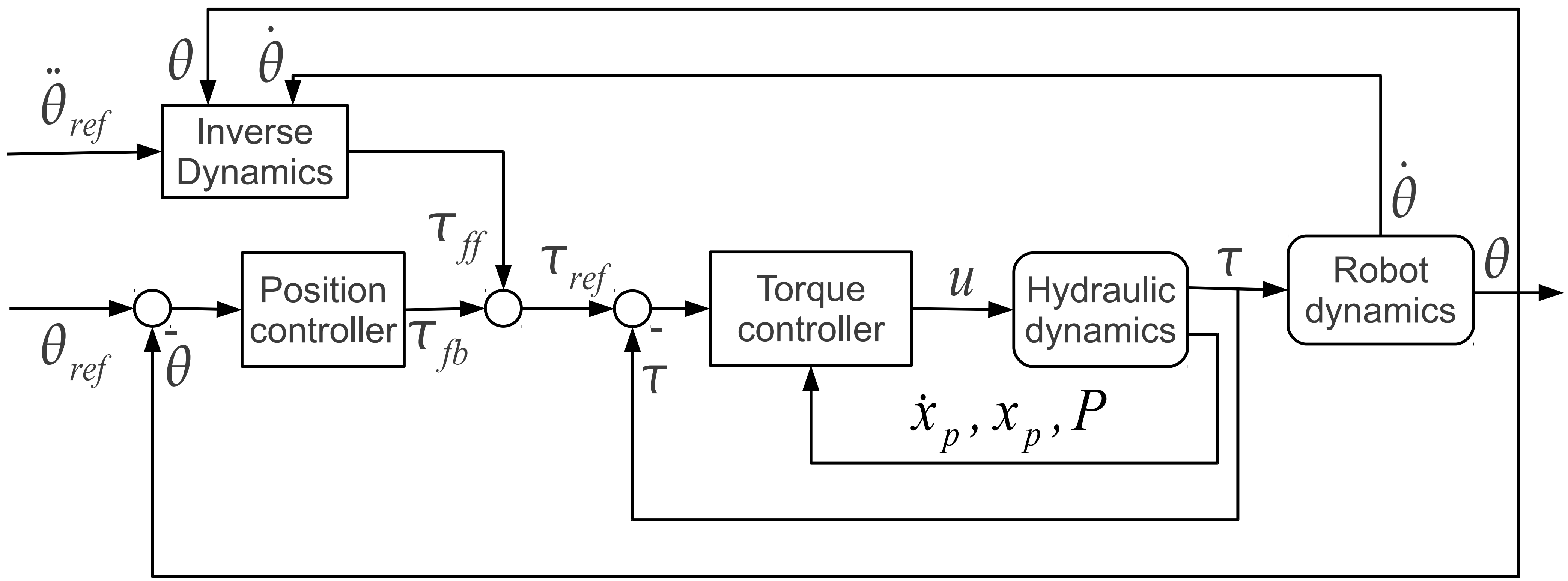
General control structure



Torque controlled robot - Control diagram



Impedance control

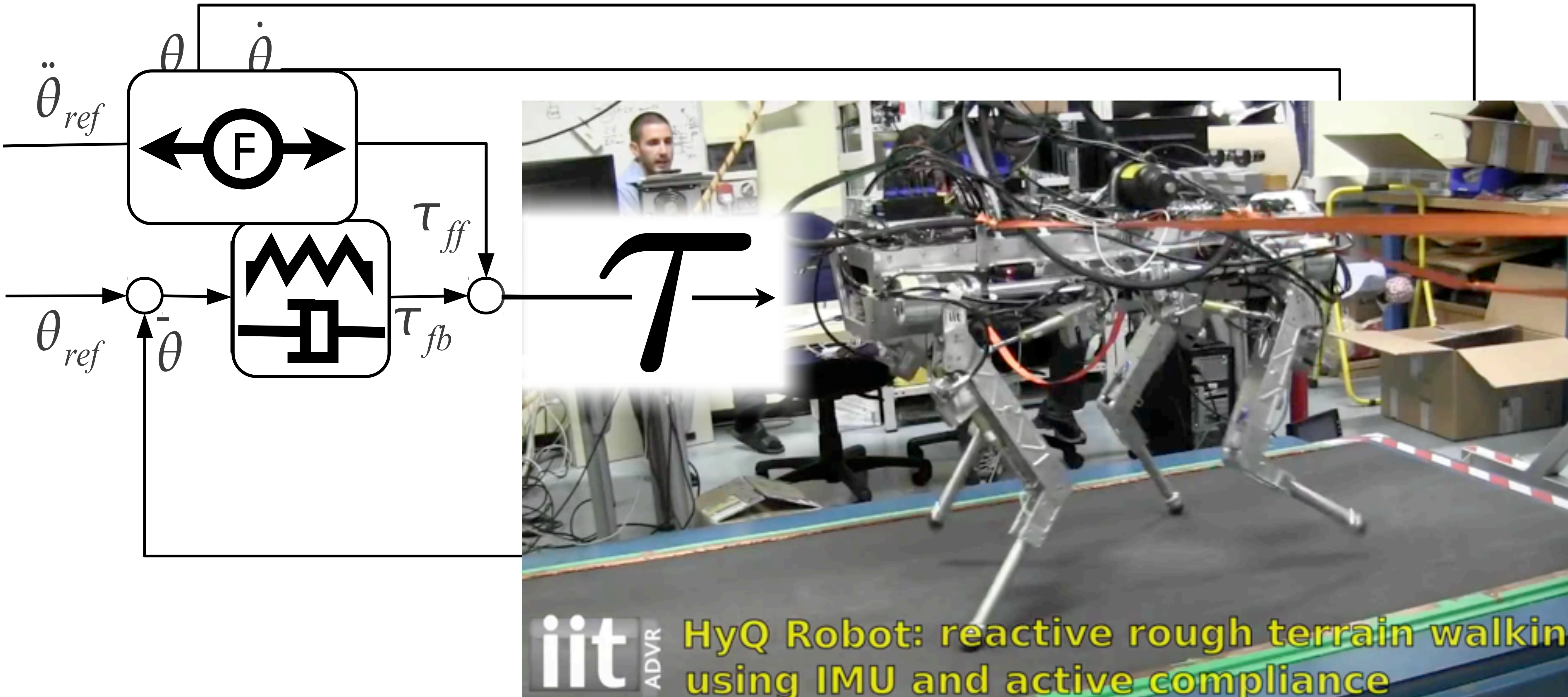


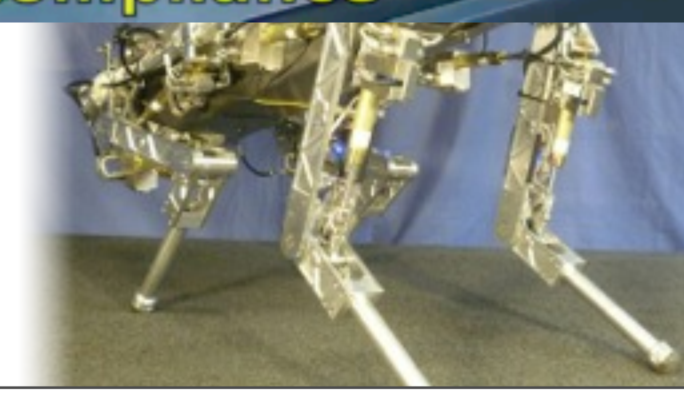
Compare to LQR control structure:

$$\mathbf{T} = -\mathbf{K}_P(\mathbf{q} - \mathbf{q}_d) - \mathbf{K}_D(\dot{\mathbf{q}} - \dot{\mathbf{q}}_d) + \mathbf{T}_{ff}$$

units?

Torque controlled legged robots



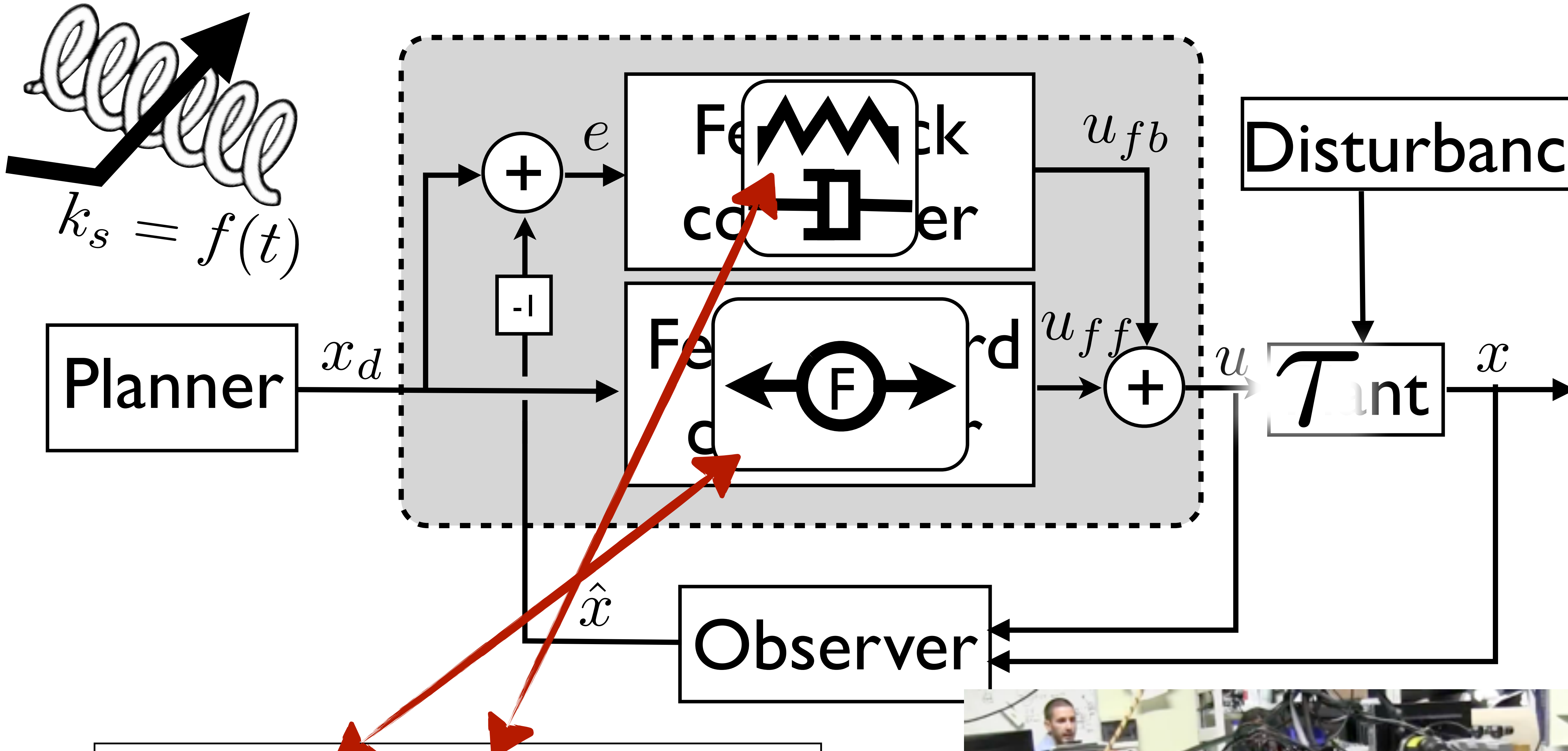


Torque control

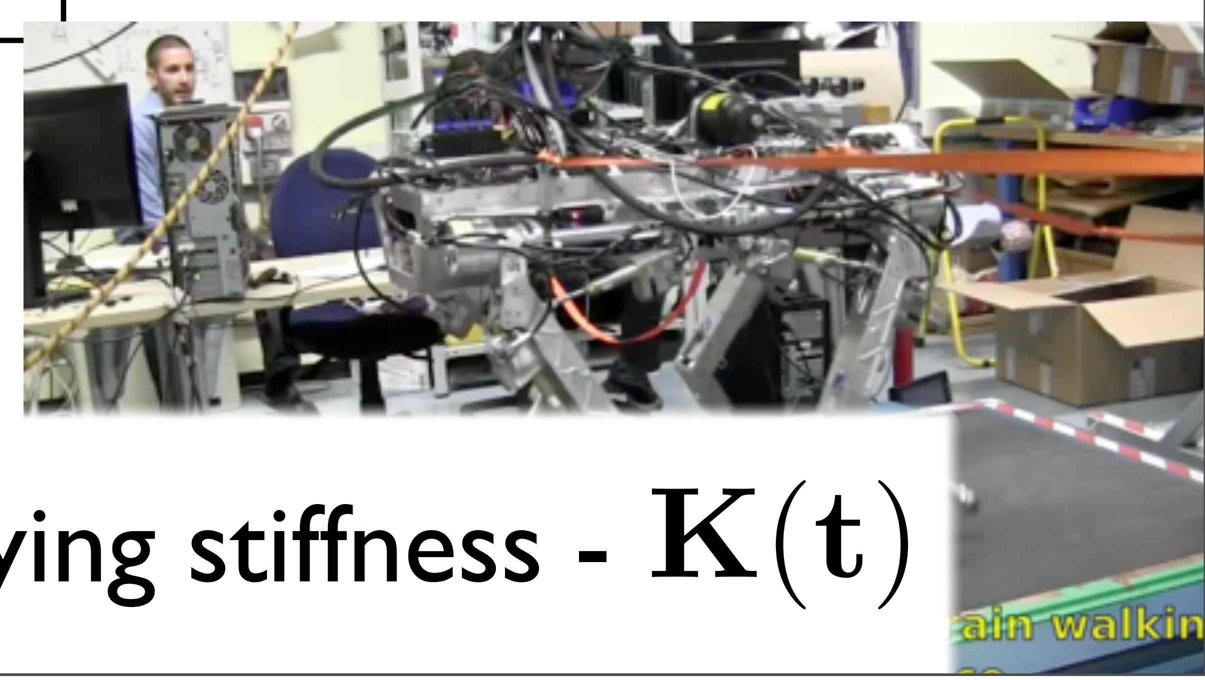
+ dynamics model

A D R L

Optimal Impedance Control

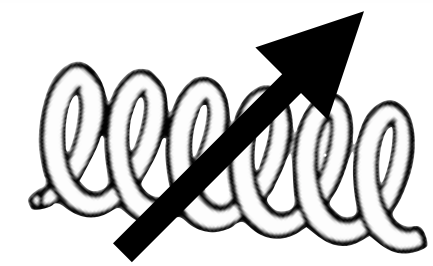


$$\mathbf{u}(t) = \mathbf{u}^*(t) + \Delta\mathbf{u}^*(t)$$
$$= \mathbf{u}^*(t) - \mathbf{C}(t)[\mathbf{x}(t) - \mathbf{x}^*(t)]$$



Principle of optimality: Time varying stiffness - $\mathbf{K}(\mathbf{t})$

(Virtual compliance

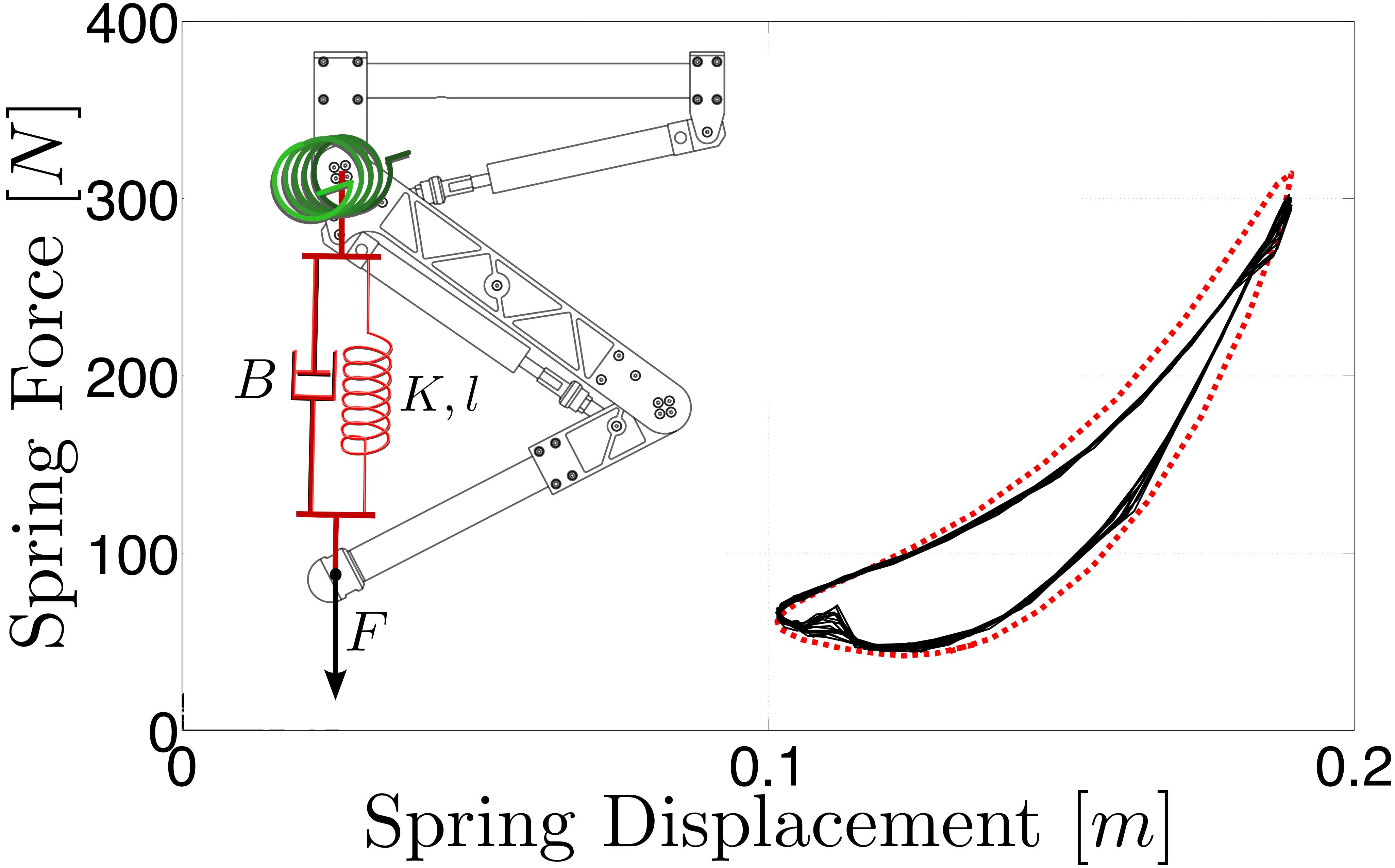


$$k_s = f(t)$$

Active springs!

Virtual nonlinear springs

$$\tau = J^T F$$



ADRL

ETH Zürich

Active vs passive compliance

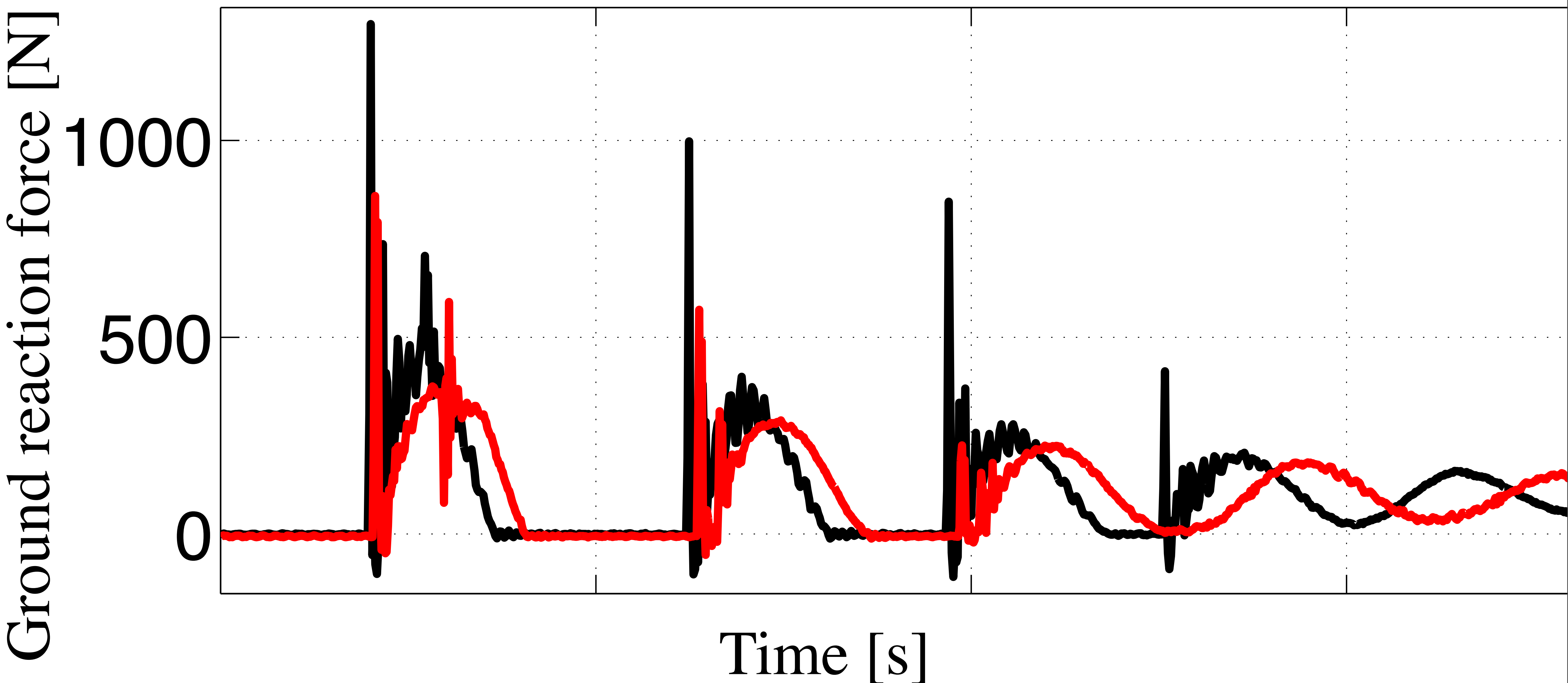
Controller: [Boaventura, ICRA 2012]

A D R L

robotics Swiss National Centre of Competence in Research

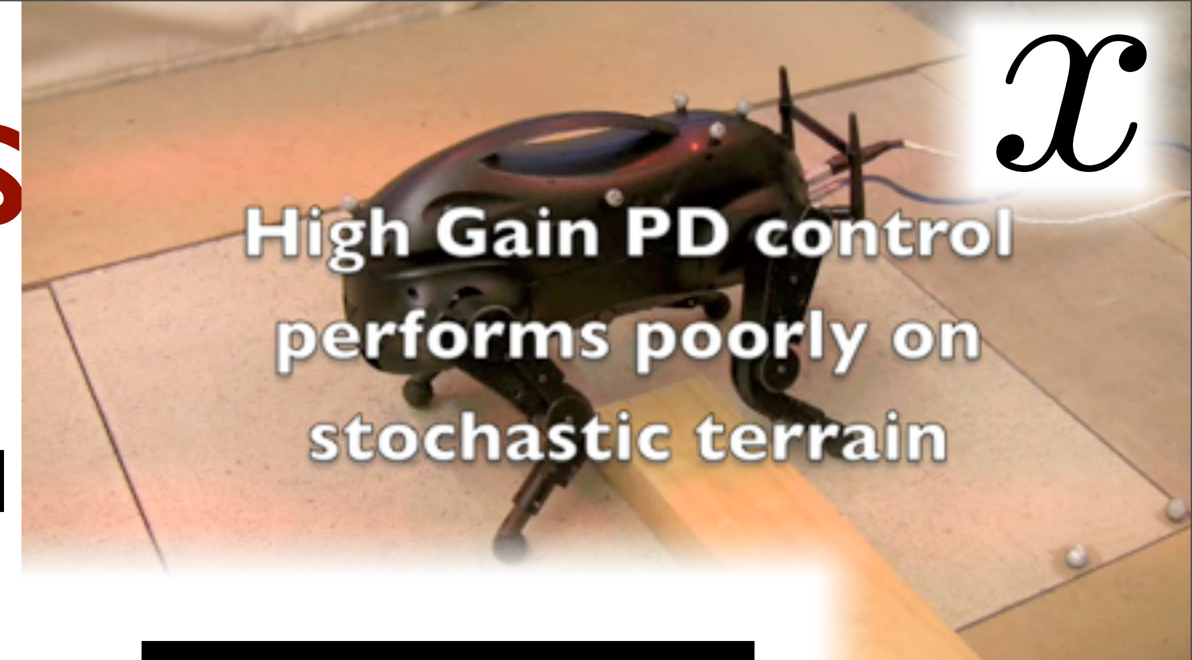
ETH Zürich

Active vs passive compliance



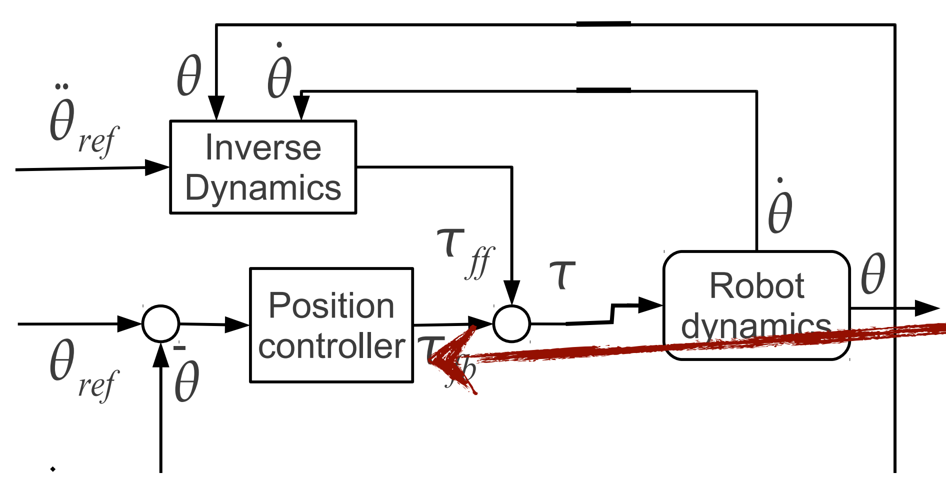
)

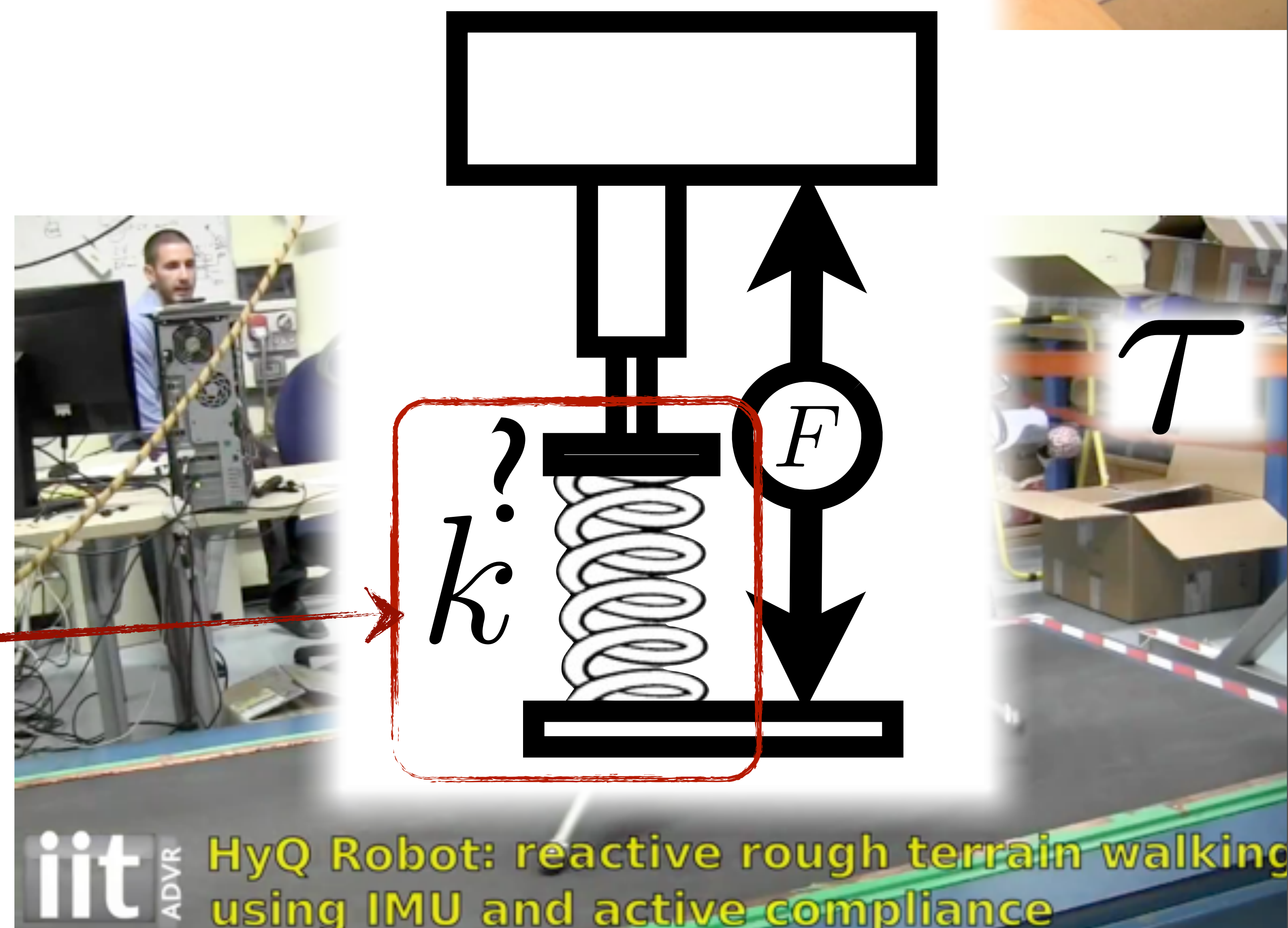
Inverse dynamics



Position control
only kinematic model

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{S}^T\tau + \mathbf{J}_C^T(\mathbf{q})\lambda$$





Torque control
+ dynamics model

Parametrization

$\Upsilon(t)^T$

Policies to be learned

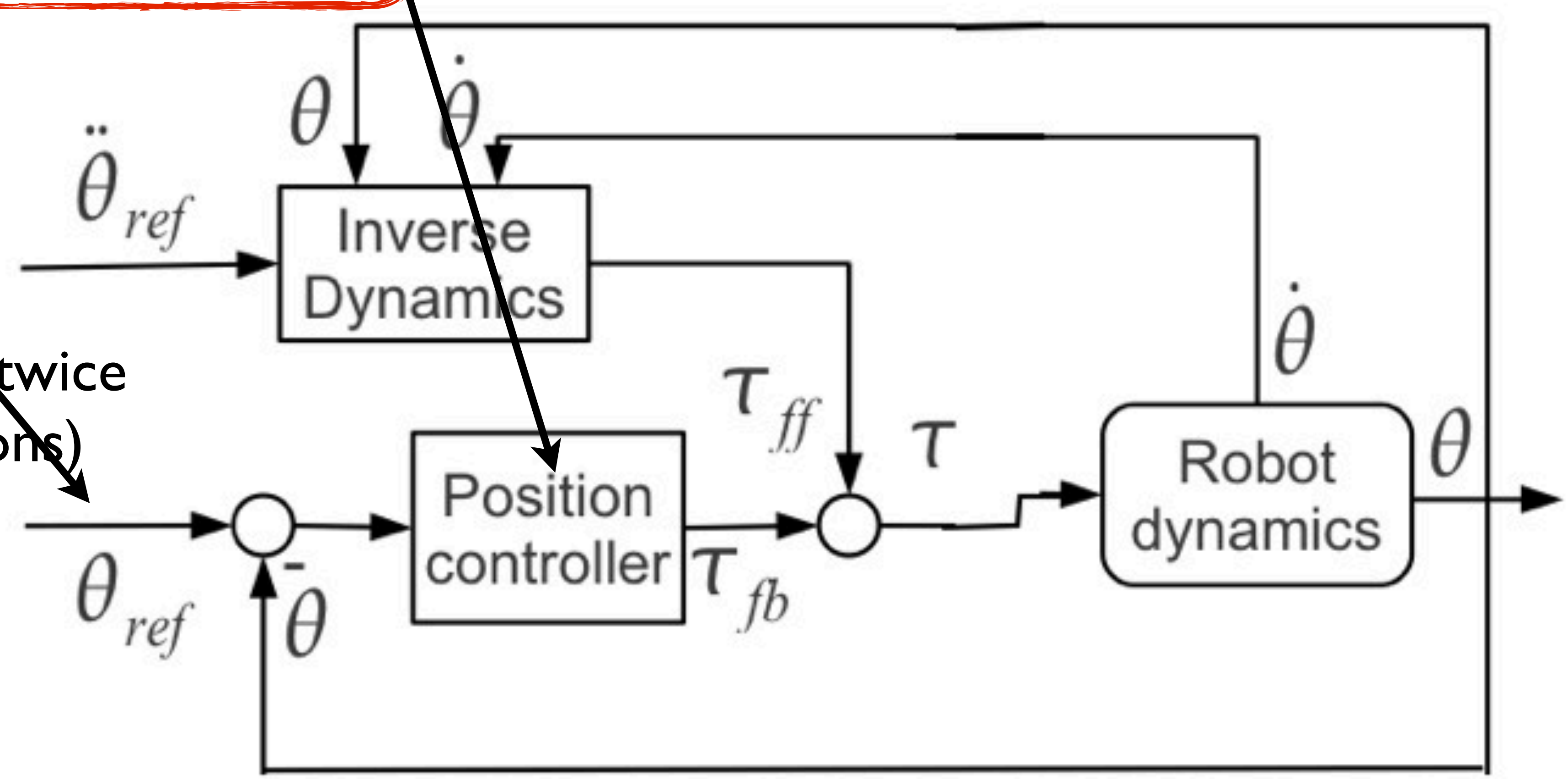
Position reference: $\mathbf{x_D(t) = \phi(t)^T \theta_x}$ — Spring resting length

Gain schedules:

$\mathbf{K_P(t) = \phi(t)^T \theta_p}$ — Spring

$\mathbf{K_D(t) = \phi(t)^T \theta_d}$ — Damper

make sure position reference is twice differentiable (finite accelerations)



A D R L

Via-point task

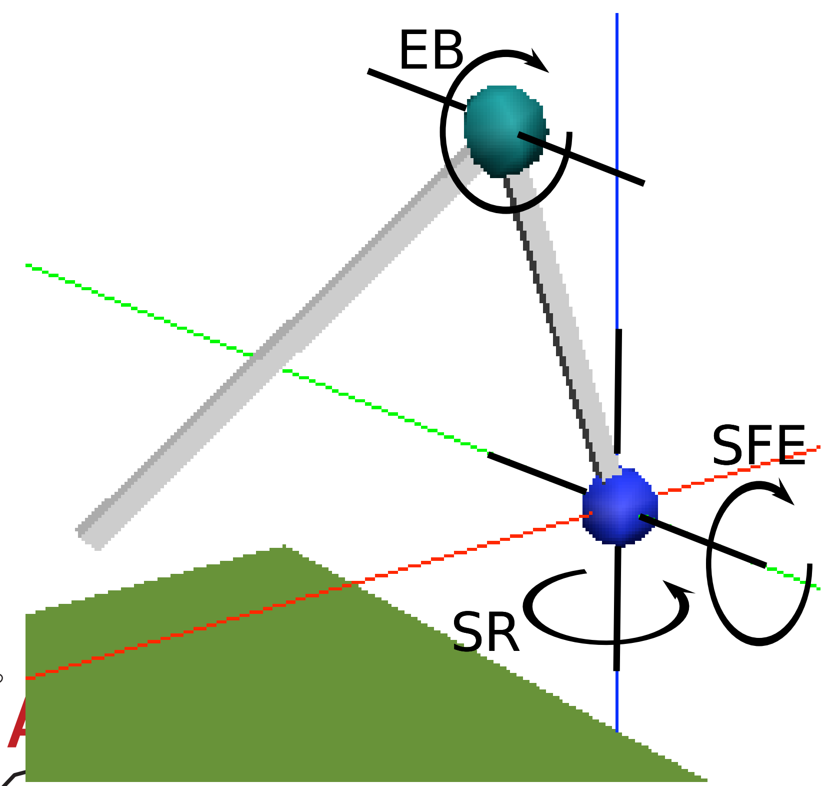
$$r_t = W_{gain}\sum_i K^i_{P,t} + W_{acc}\|\ddot{\mathbf{x}}\| + W_{via-point}C(t)$$

'keep gains low / stay soft!

'don't wiggle too much'

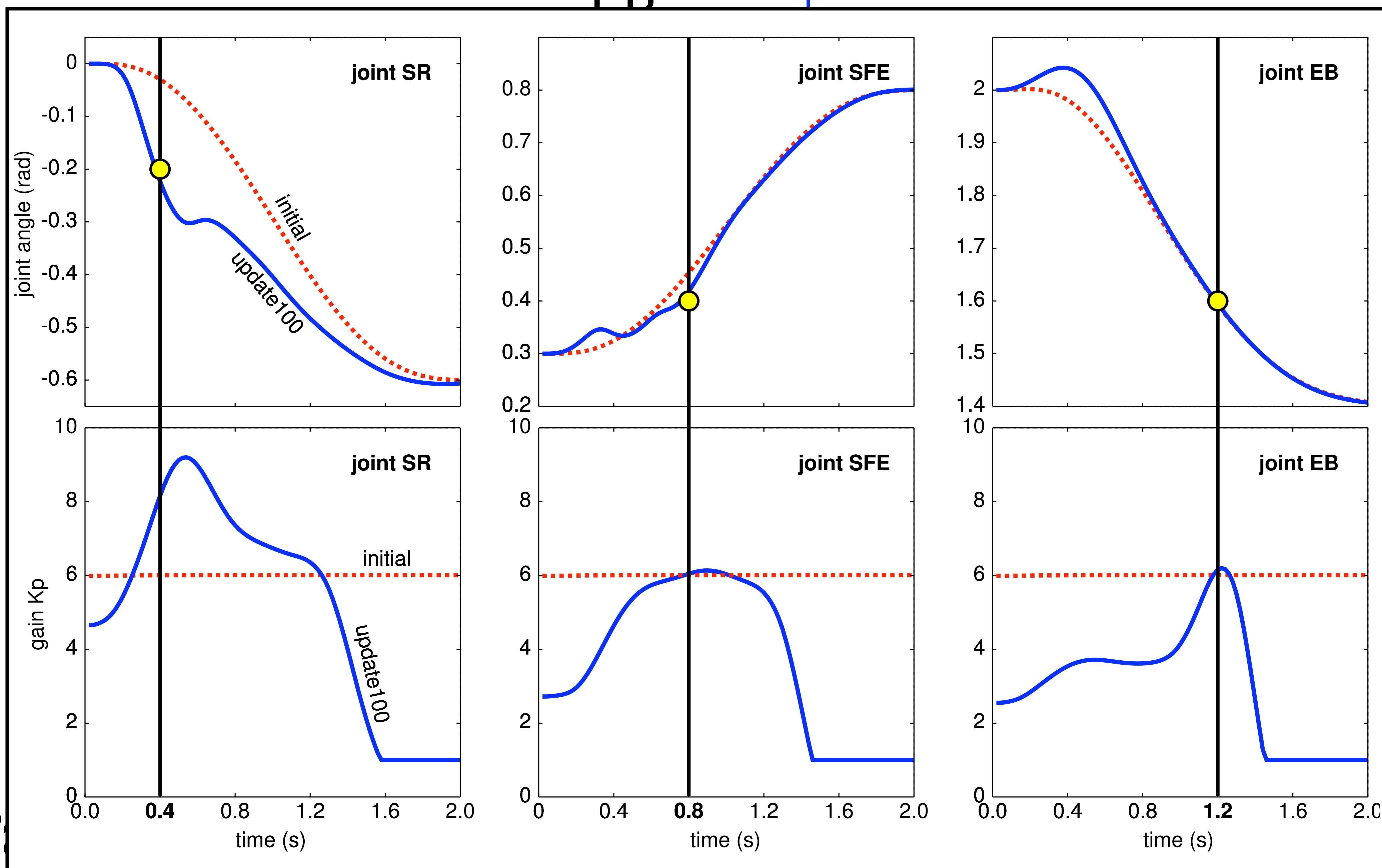
go through given joint angles at given times:

$$C(t) = \delta(t-0.4)\cdot|\, q_{SR}(t) + 0.2\,| + \delta(t-0.8)\cdot|\, q_{SFE}(t) - 0.4\,| + \delta(t-1.2)\cdot|\, q_{EB}(t) - 1.5\,|$$



Gain scheduling

3 DOF - Phantom robot



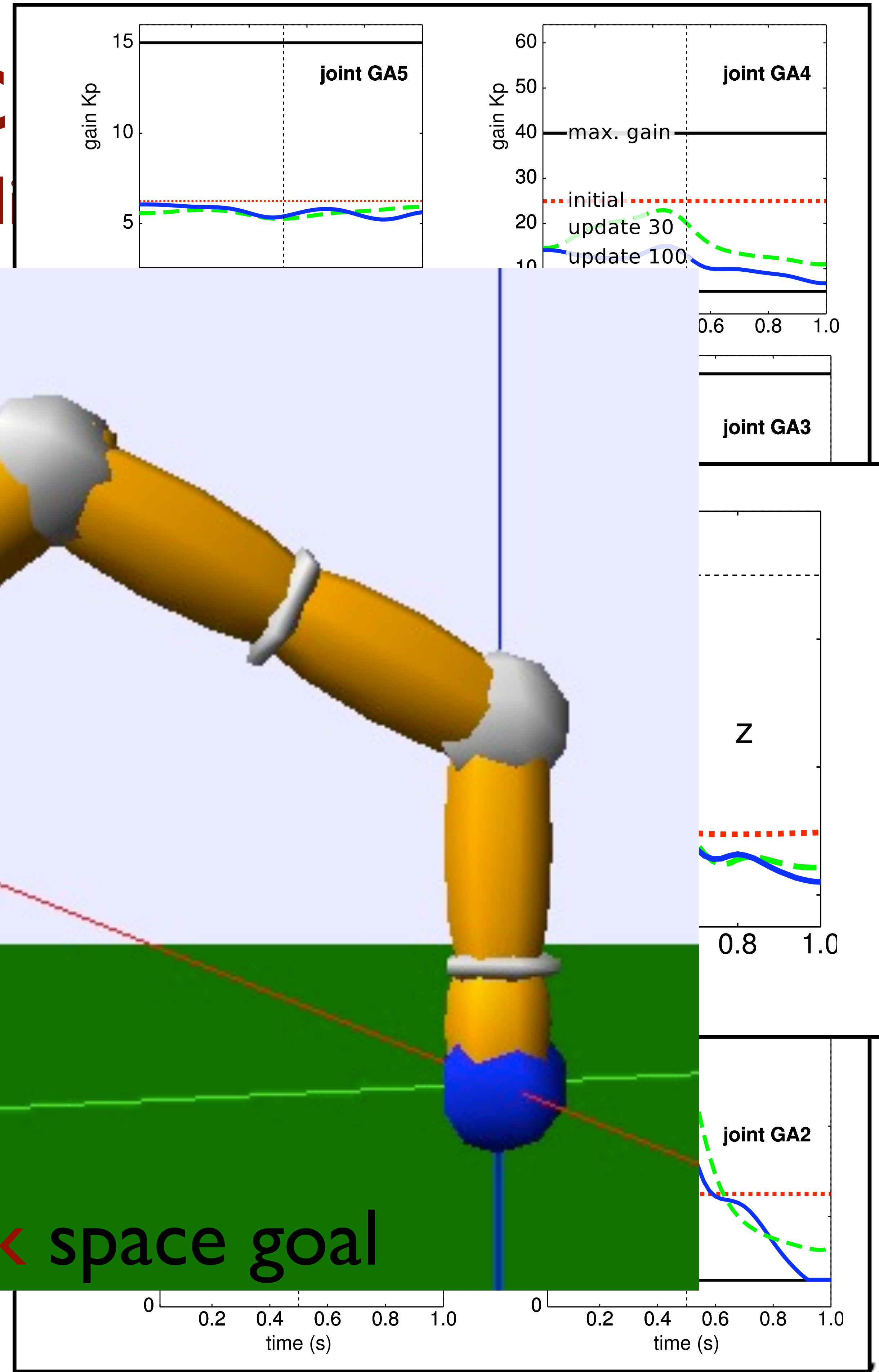
[Buchli et al, RSS 2010]

ETH Zürich

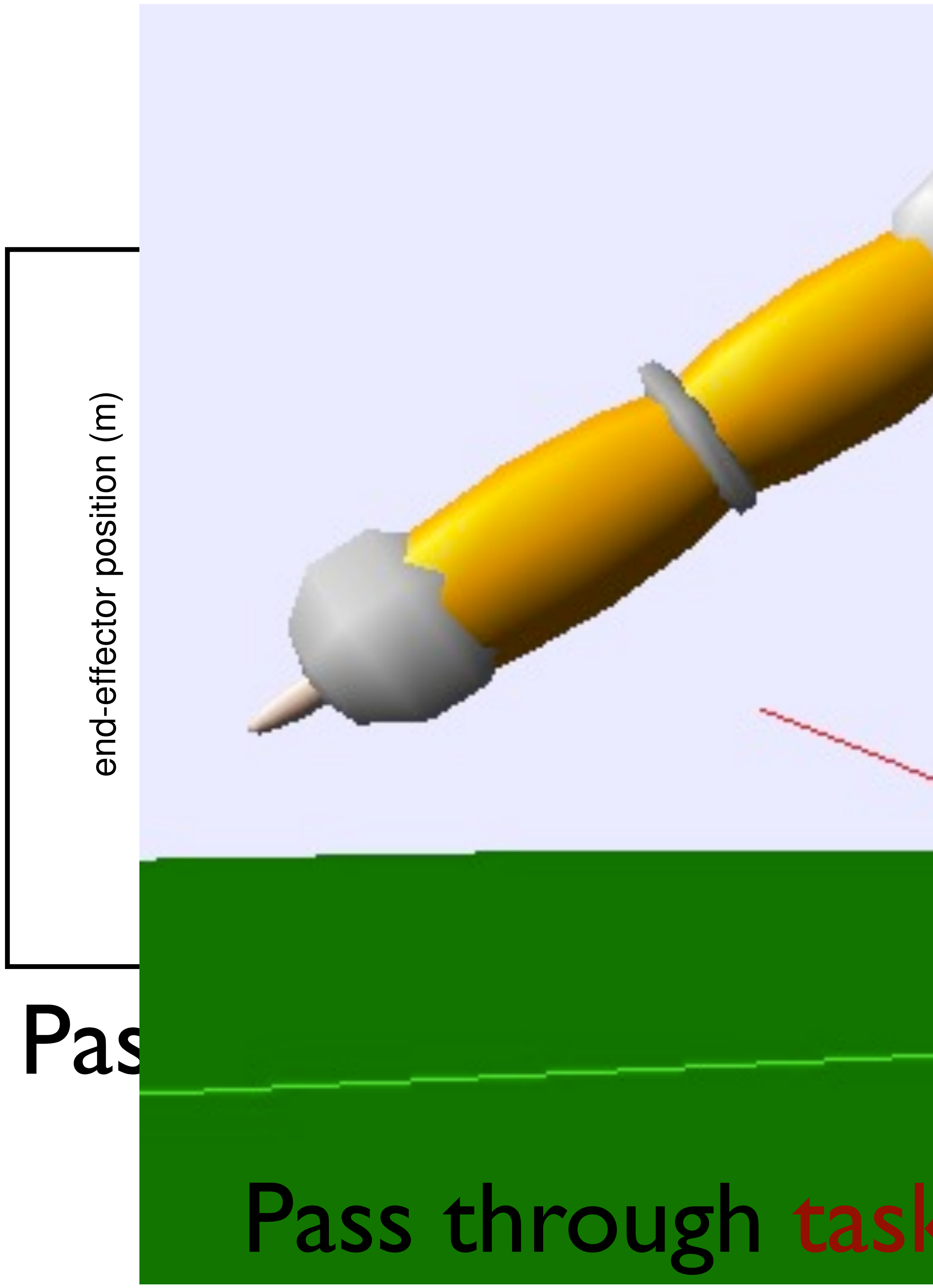
[Buchli et al, RSS 2010]

Gain sc

6DOF - Kuka li



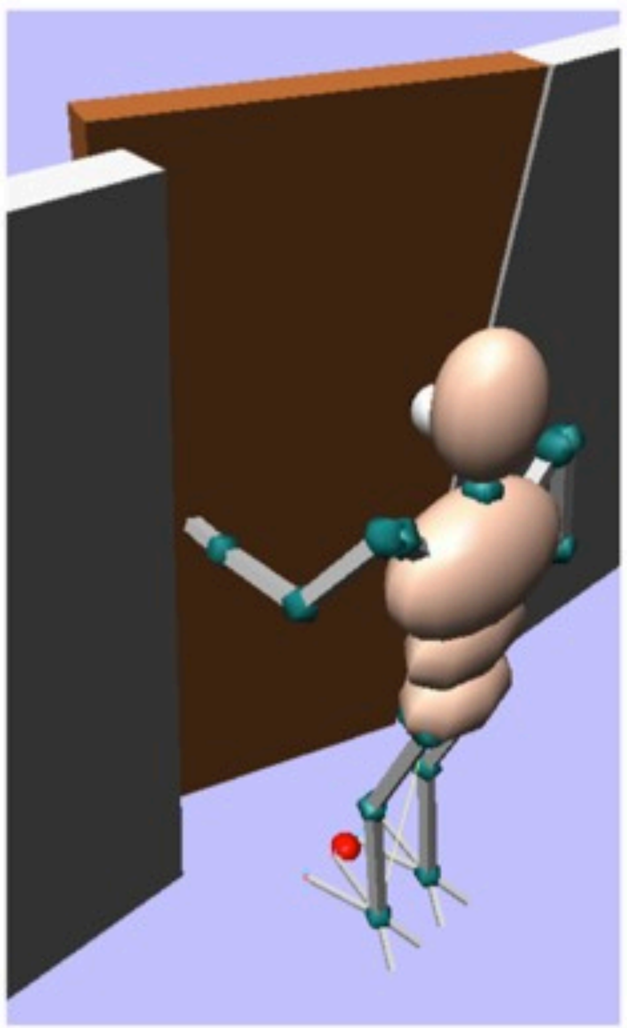
Pas

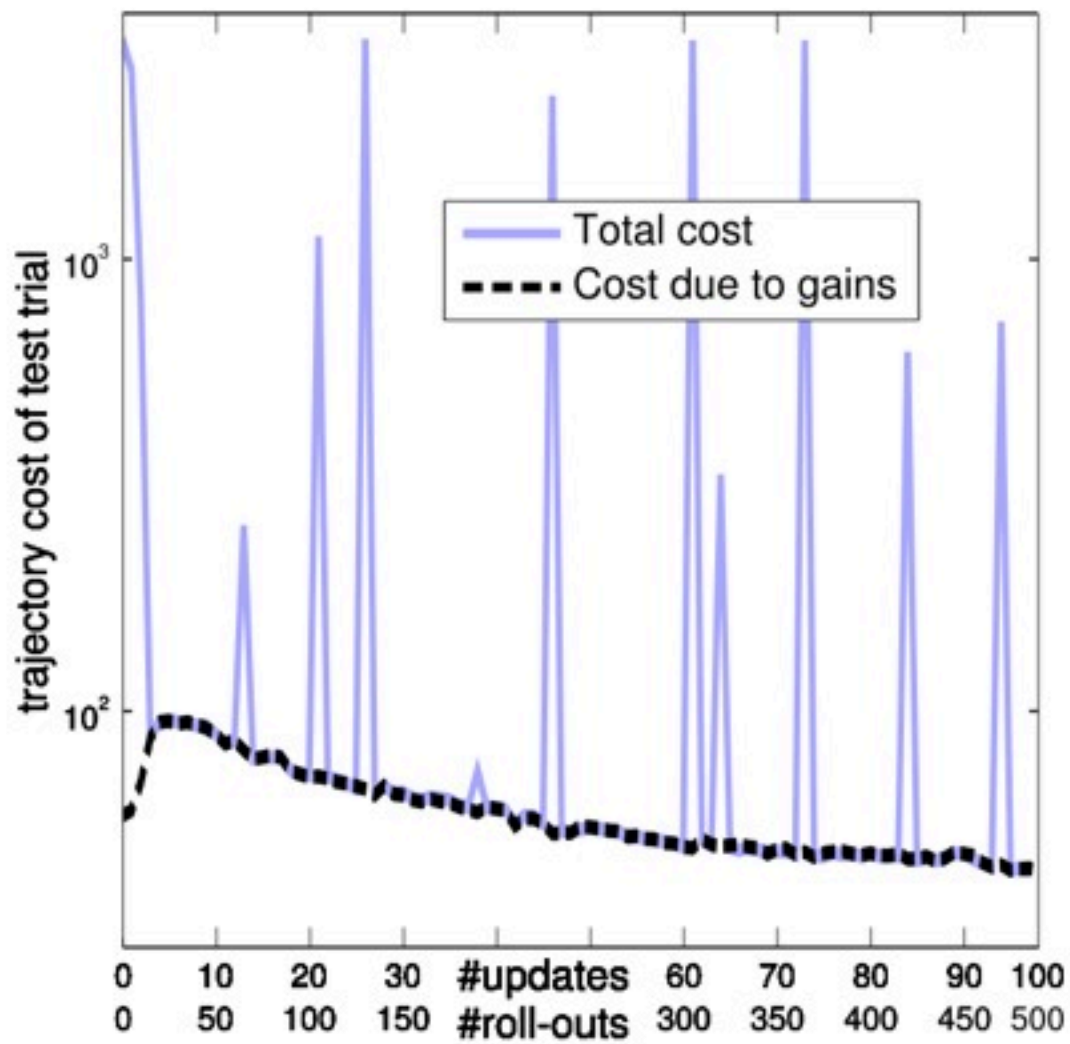


A D R L

Zürich

Goal: Open door as far as possible and use lowest amount of effort

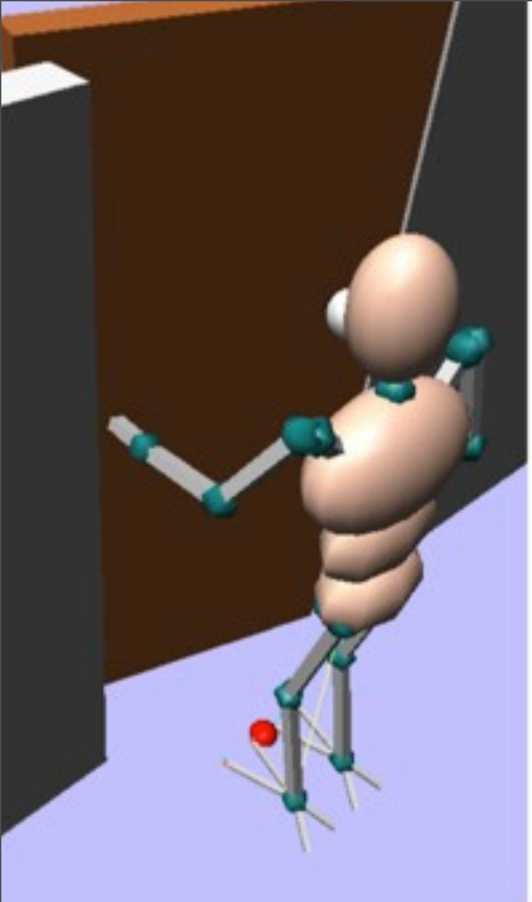




$$r_t = \frac{1}{N} \sum_{i=1}^{3} K_{P,t}^{i}$$

$$\phi_{t_N} = 10^4 \cdot (\psi_{max} - \psi_N)$$

A D R L



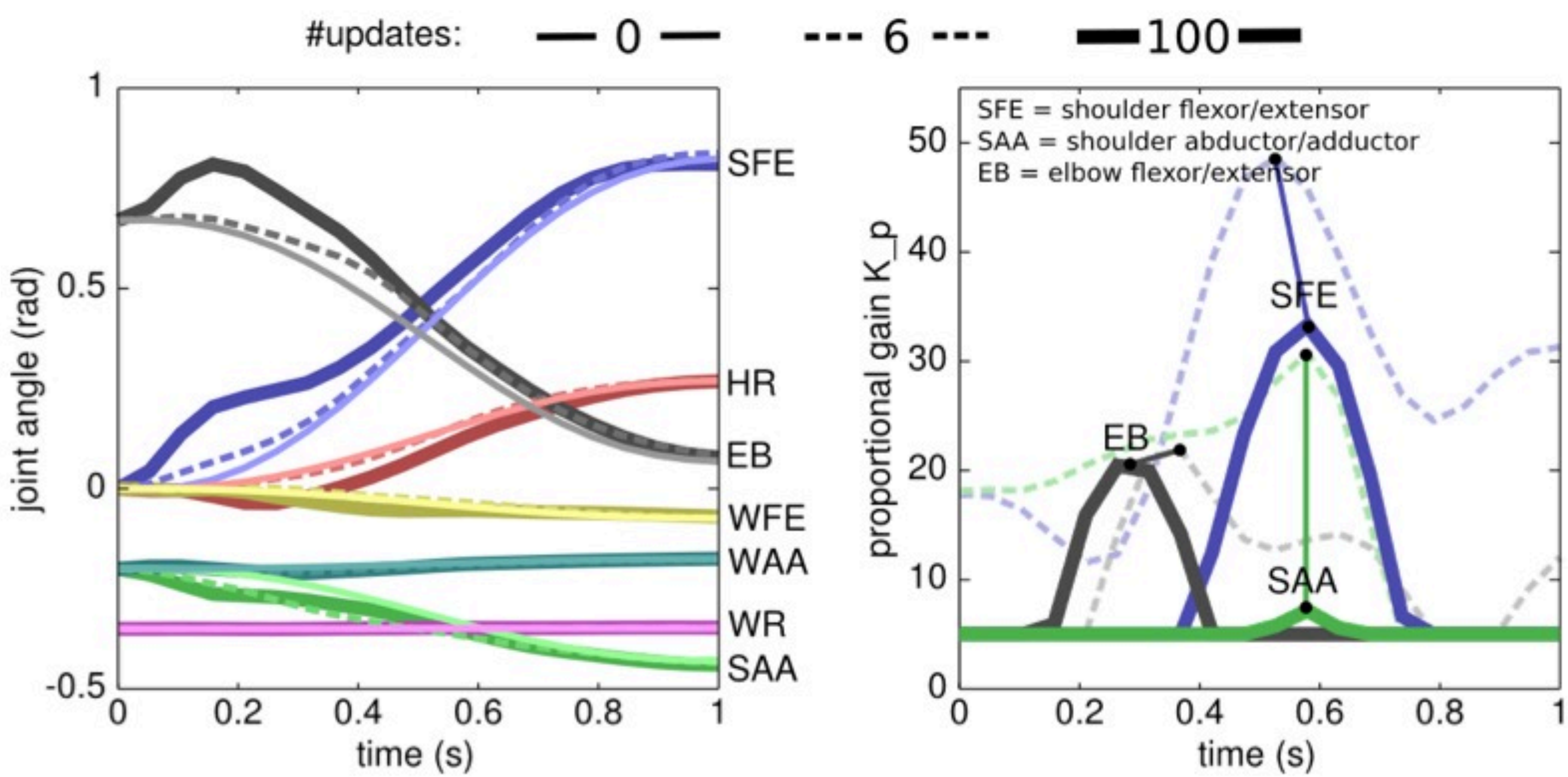
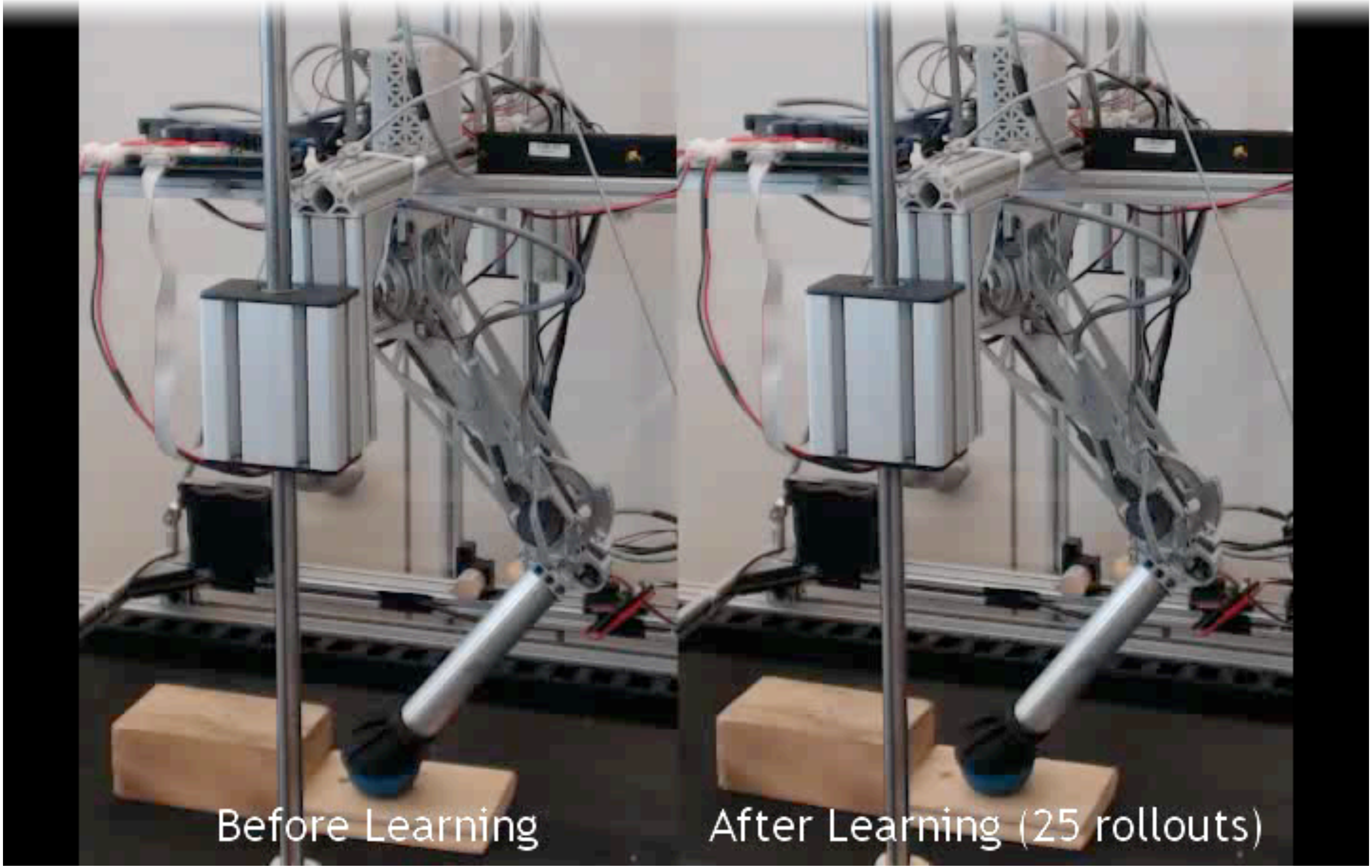


Fig. 4. Learned joint angle trajectories (center) and gain schedules (right) of the CBi arm after 0/6/100 updates. The gain schedules of only three joints have been depicted for sake of clarity.

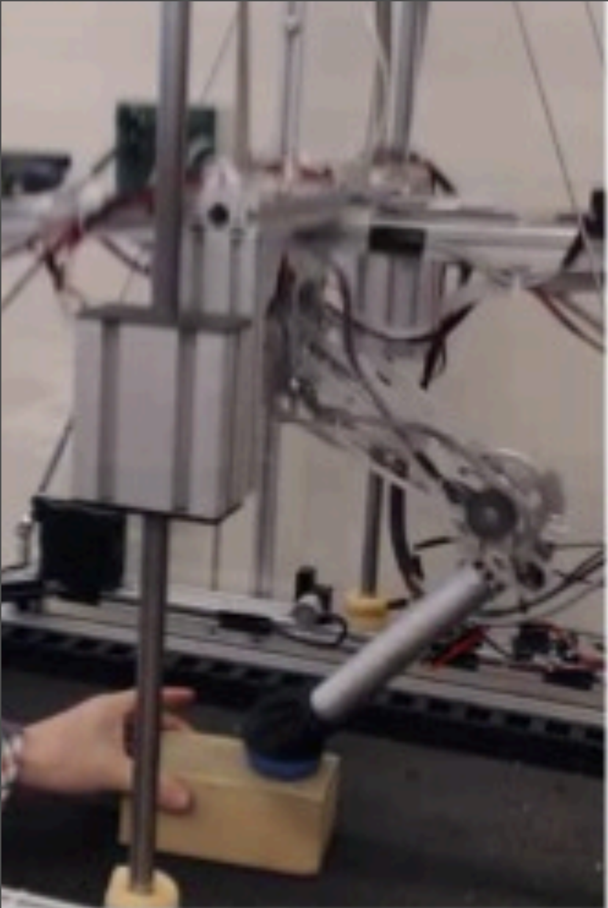
ADRL

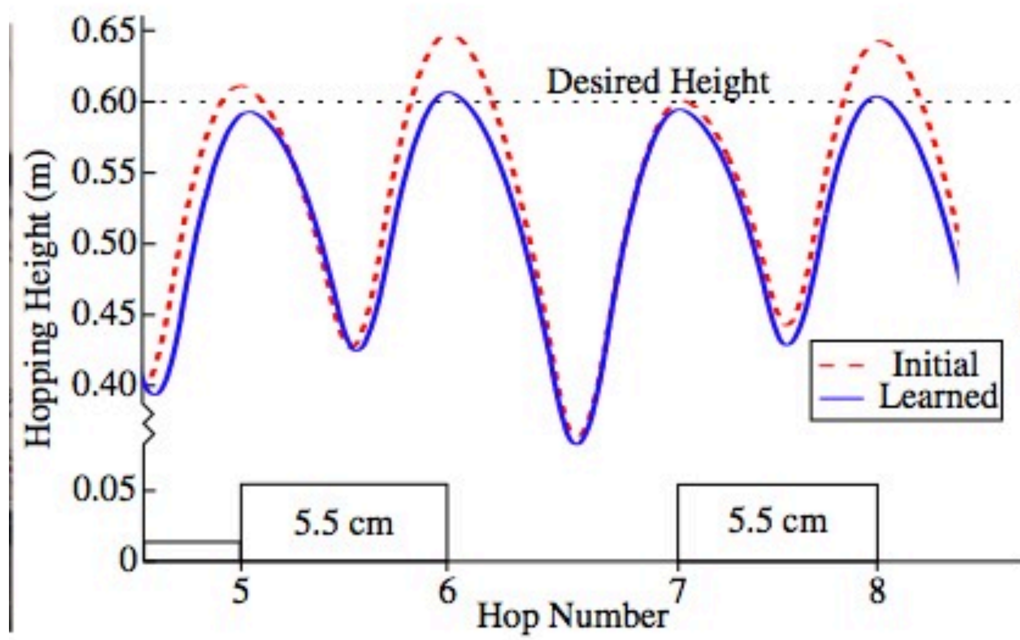
ETH Zürich



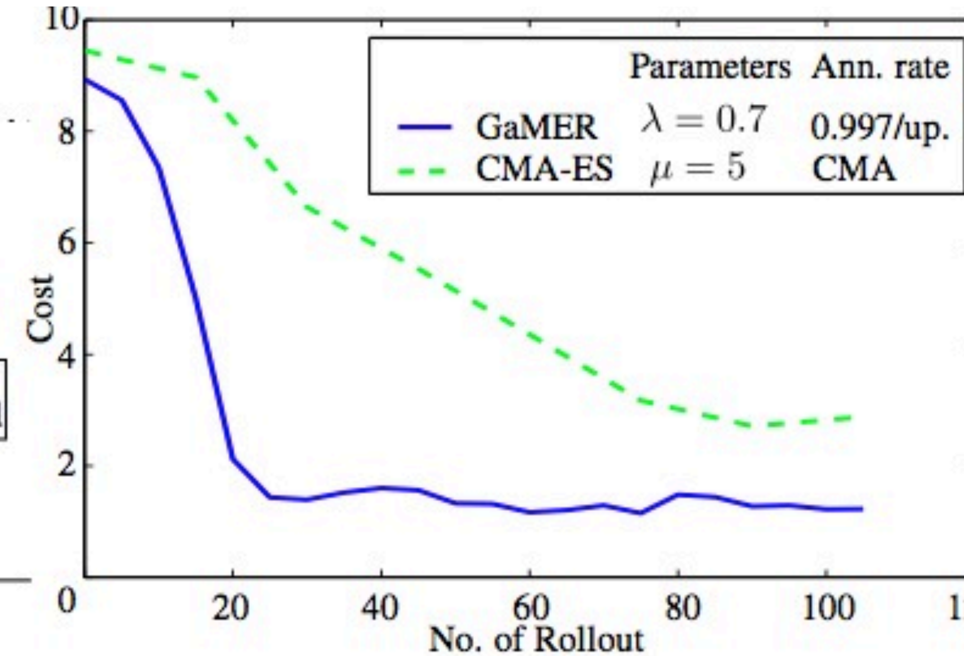
ADRL

ETH Zürich

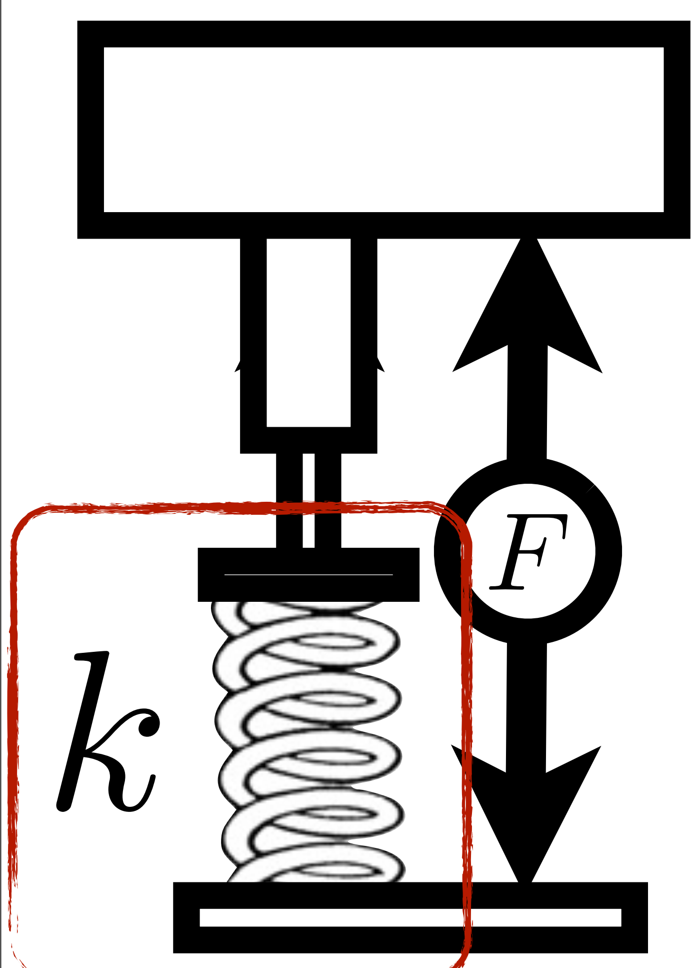




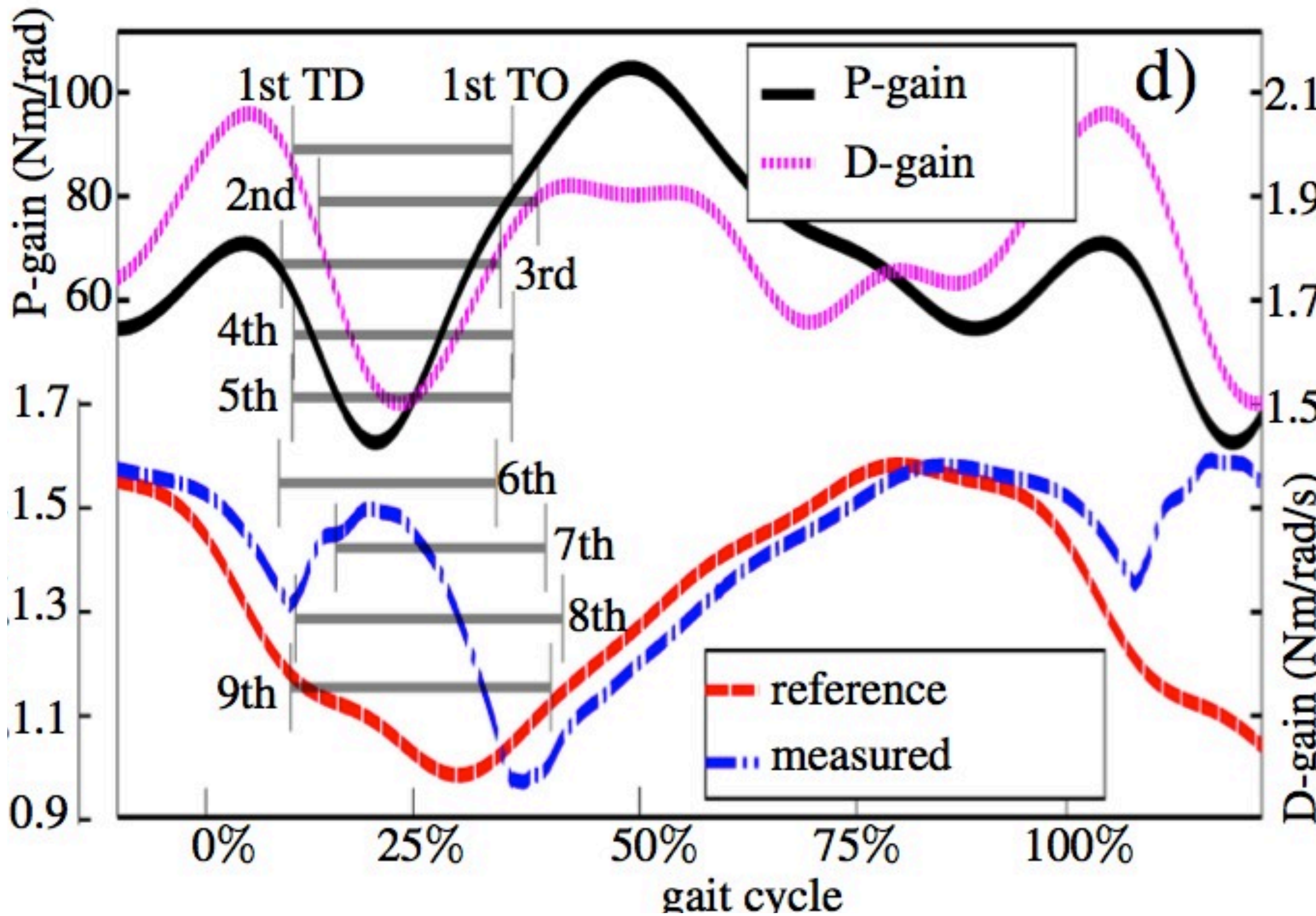
Desired Height
Hopping Height (m)
0.65
0.60
0.55
0.50
0.45
0.40
0.05
0
Initial
Learned
5.5 cm
5.5 cm
5
6
7
8
Hop Number



Parameters Ann. rate
GaMER $\lambda = 0.7$ 0.997/up.
CMA-ES $\mu = 5$ CMA
Cost
10
8
6
4
2
0
20
40
60
80
100
120
No. of Rollout



F
k



d)
P-gain
D-gain
P-gain (Nm/rad)
100
80
60
1st TD
1st TO
2nd
3rd
4th
5th
6th
7th
8th
9th
2.1
1.9
1.7
1.5
1.7
1.5
1.3
1.1
0.9
reference
measured
D-gain (Nm/rad/s)
0%
25%
50%
75%
100%
gait cycle

h

PI^2 - General Algorithm

PI2 for state feedback control

'feedforward' 'state-feedback'

$$u_i(t,\mathbf{x}) = \text{grand sum}\left[\bar{\mathbf{\Upsilon}}(t,\mathbf{x}) \circ \boldsymbol{\theta}_i\right]$$

$$\bar{\mathbf{\Upsilon}}(t,\mathbf{x}) = \mathbf{\Upsilon}(t)\begin{bmatrix}1 & \mathbf{x}^T\end{bmatrix} = \begin{bmatrix} e^{-\frac{1}{2}\frac{(t-\mu_n)^2}{\sigma_n^2}} \begin{bmatrix}1 & \mathbf{x}^T\end{bmatrix}\end{bmatrix}_{N\times(1+\dim[\mathbf{x}])}$$

$\boldsymbol{\theta}_i$ is a $N\times(1+\dim[\mathbf{x}])$ parameter matrix for ith control input approximation

$\bar{\mathbf{\Upsilon}}(t,\mathbf{x})$ basis function matrix $N\times(1+\dim[\mathbf{x}])$

Policy nonlinear in time, linear in states

Algorithm 10 General PI2 Algorithm

general policy: parameter matrix

given

The cost function:

$J = \Phi(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} \left(q(t,\mathbf{x}) + \frac{1}{2}\mathbf{u}^T\mathbf{R}\mathbf{u}\right) dt$

A Linear Model for function approximation: $\mathbf{u}(t,\mathbf{x}) = [u_i(t,\mathbf{x})] = [\text{grand sum}[\bar{\mathbf{\Upsilon}}(t,\mathbf{x}) \circ \boldsymbol{\theta}_i]]$

Initialize $\{\boldsymbol{\theta}_i\}$ with a sophisticated guess

Initialize exploration noise standard deviation: c

repeat

Create K rollouts of the system with the perturbated parameter $\{\boldsymbol{\theta}_i\}+\{\boldsymbol{\epsilon}_i\}$, $\{\boldsymbol{\epsilon}_{i,j}\} \sim \mathcal{N}(\mathbf{0}, c^2\mathbf{I})$

for the ith control input **do**

for each time, s **do**

Calculate the Return from starting time s for the kth rollout:

$R(\tau^k(s)) = \Phi(\mathbf{x}(t_f)) + \int_s^{t_f} \left(q(t,\mathbf{x}) + \frac{1}{2}\mathbf{u}^T\mathbf{R}\mathbf{u}\right) dt$

Calculate α from starting time s for the kth rollout:

$\alpha^k(s) = \exp(-\frac{1}{\lambda}R(\tau^k(s)))/\sum_{k=1}^{K}\exp(-\frac{1}{\lambda}R(\tau^k(s)))$

Calculate the time varying parameter increment $\Delta\boldsymbol{\theta}_i(s)$:

$\Delta\boldsymbol{\theta}_i(s) = \sum_{k=1}^{K} \alpha^k(s) \frac{\mathbf{\Upsilon}(s)\mathbf{\Upsilon}^T(s)}{\mathbf{\Upsilon}^T(s)\mathbf{\Upsilon}(s)} \boldsymbol{\epsilon}_i^k(s)$

end for

for the jth column of $\Delta\boldsymbol{\theta}_i$ matrix, $\Delta\boldsymbol{\theta}_{i,j}$ **do**

Time-averaging the parameter vector

$$\Delta\boldsymbol{\theta}_{i,j} = \left(\int_{t_0}^{t_f} \Delta\boldsymbol{\theta}_{i,j}(s) \circ \mathbf{\Upsilon}(s) ds\right) \cdot / \int_{t_0}^{t_f} \mathbf{\Upsilon}(s) ds$$

end for

Update parameter vector for control input i, $\boldsymbol{\theta}_i$:

$\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i + \omega\Delta\boldsymbol{\theta}_i$

end for

- Decrease c for noise annealing

until maximum number of iterations

A D R L

ETH Zürich

Combining optimal and learning control

Task specification → **Model based optimal control** → **Learning control** → Optimal *& robust* motion control on real system

Domain knowledge
'optimal initial guess'

Refinement and
adaptation to real system

quick convergence

less restrictions on cost and system dynamics

Example

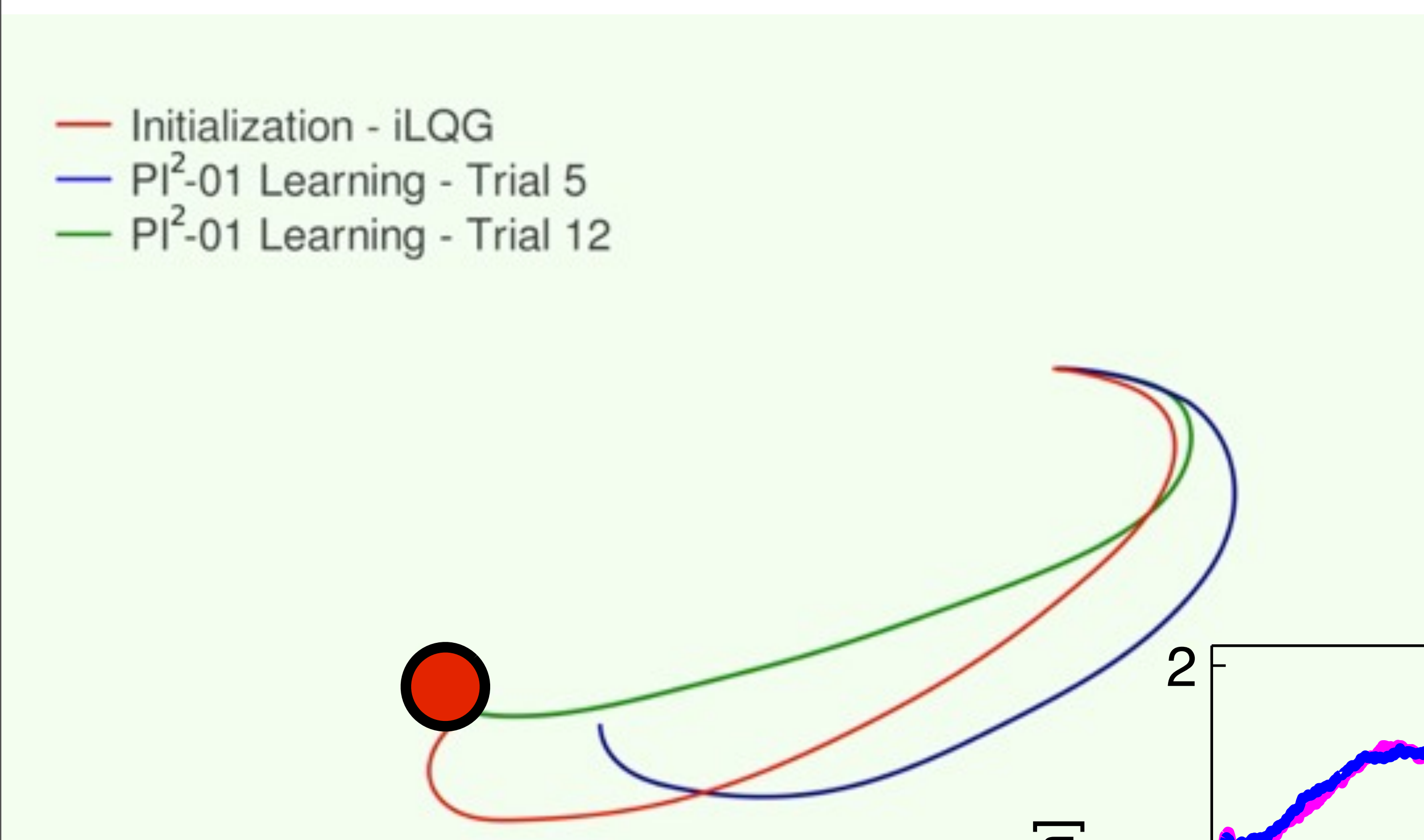
Rezero

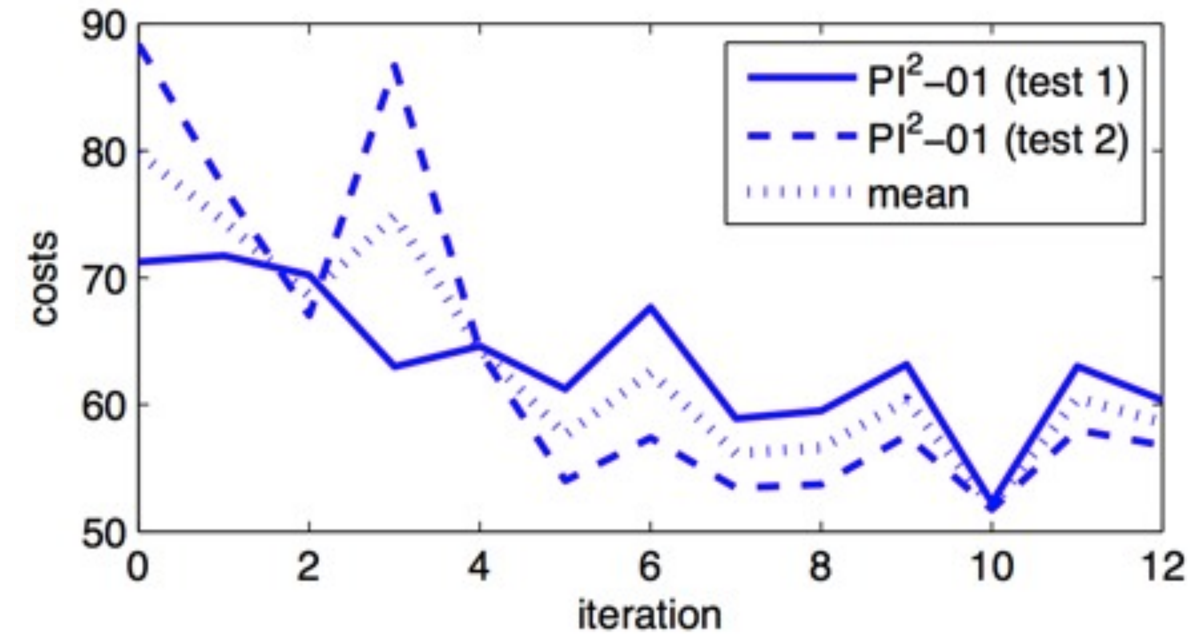
Task: ‘reach goal, using minimum torque’

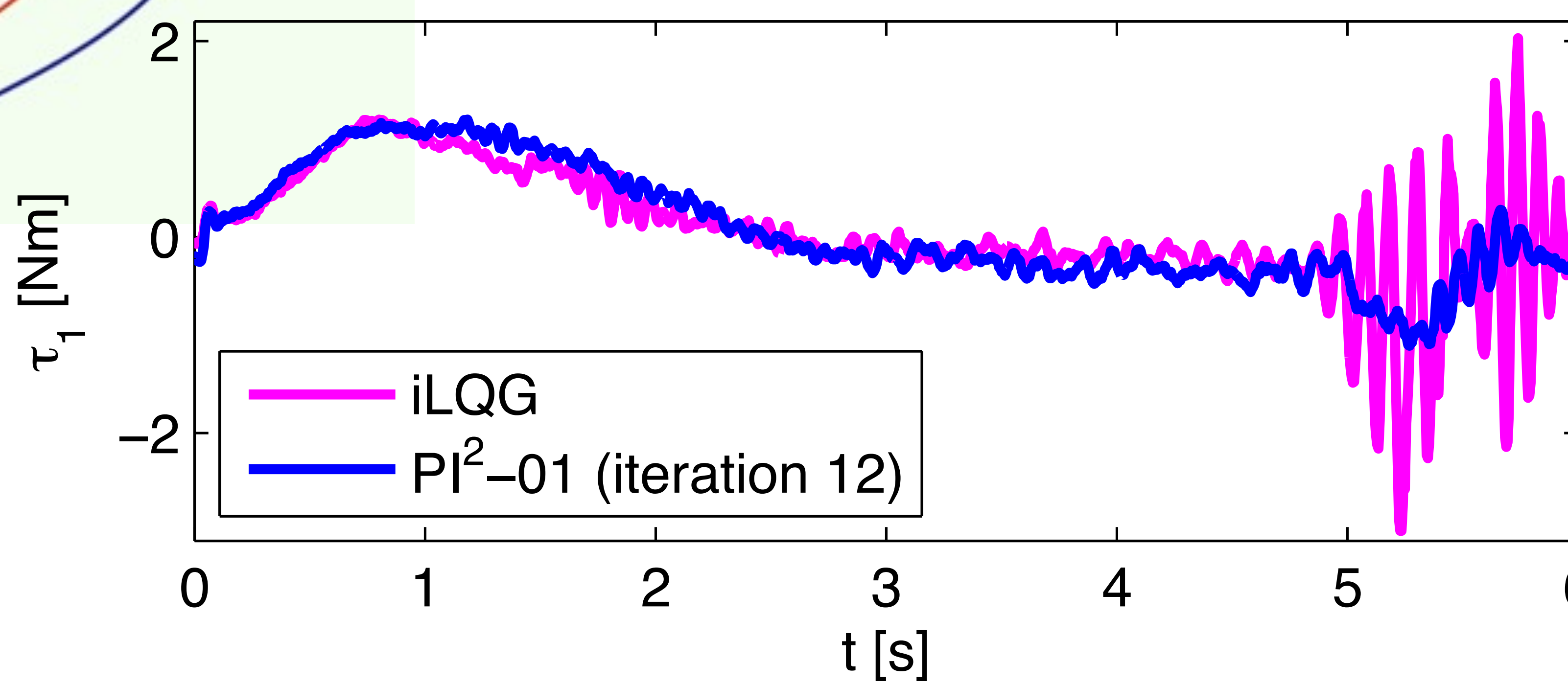
Init: iLQG   Learning: PI^2-01

[Li & Todorov, 2005]

[Farshidian, Neunert, Buchli, IROS 14]







Cost: 'reach goal, w. min. torque'

Init: iLQG Learning: PI^2-01

A D R L

ETH Zürich