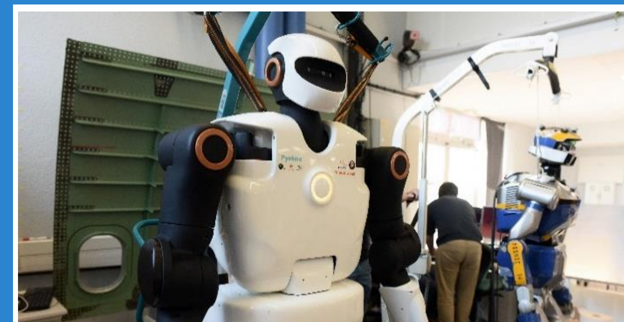
memmo

European Commission | Horizon 2020 European Union funding for Research & Innovation

Introduction to Optimal Control



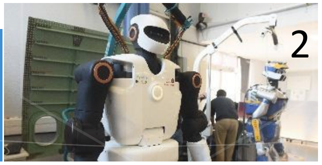
Ludovic Righetti (NYU)

Nicolas Mansard (CNRS)

cnrs idiap The University of Edinburgh Max-Planck-Gesellschaft University of Oxford PAL Robotics Wandercraft Costain Airbus APAJH

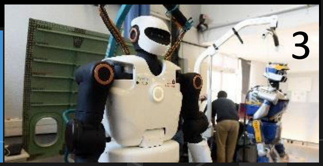
Outline

- What can we do with optimal control?
- Where is optimal-control is the robot galaxy?
- What is dynamic programming?
- Should you shoot or collocate?
- Why make your dynamic program *differential*?
- Is multiple shooting about guns?
- What is Crocoddyl good for, and what is beyond?



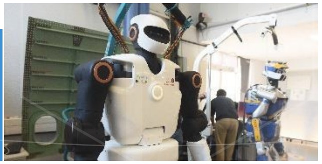
What can we do with optimal control?

VIDEO INTRODUCTION



Autonomous Driving

Information Theoretic Model
Predictive Control
[Williams et al. 2018]

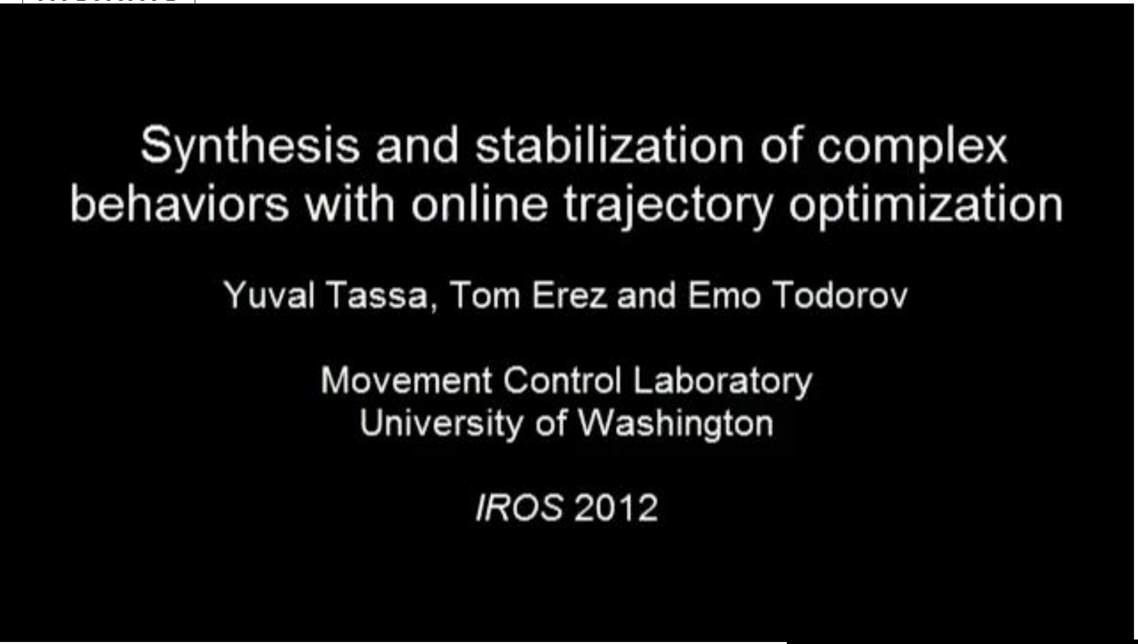


Legged Locomotion

OC with Linear Inverted Pendulum Model
[Herdt et al. 2010]

OC with Centroidal Momentum Dynamics and Full Body Kinematics
[Ponton et al. 2018], [Carpentier et al. 2018], [Dai et al. 2014], [Herzog et al. 2015]

Full-body Optimal Control



[Tassa et al. 2010]
DDP with Full-Body Dynamics
(realtime control)

[Mordatch et al. 2012]
Nonlinear Optimization for
Multi-Contact Tasks

What is dynamic programing

INTRODUCTION TO BELMAN'S EQUATIONS

Optimal control problem

$$\min_{u_0, \cdots, u_{T-1}} \sum_{t=0}^{T-1} l_t(x_t, u_t) + l_T(x_T)$$

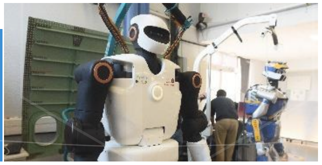
Find control inputs to minimize cost

stage costs

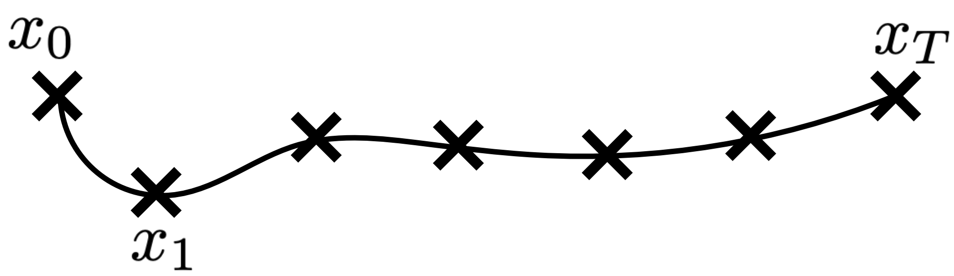
terminal cost

$$x_{t+1} = f_t(x_t, u_t)$$ deterministic dynamics

$$g(x_t, u_t) \leq 0$$ state and control constraints

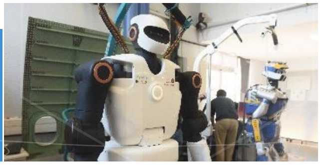


Optimal control problem

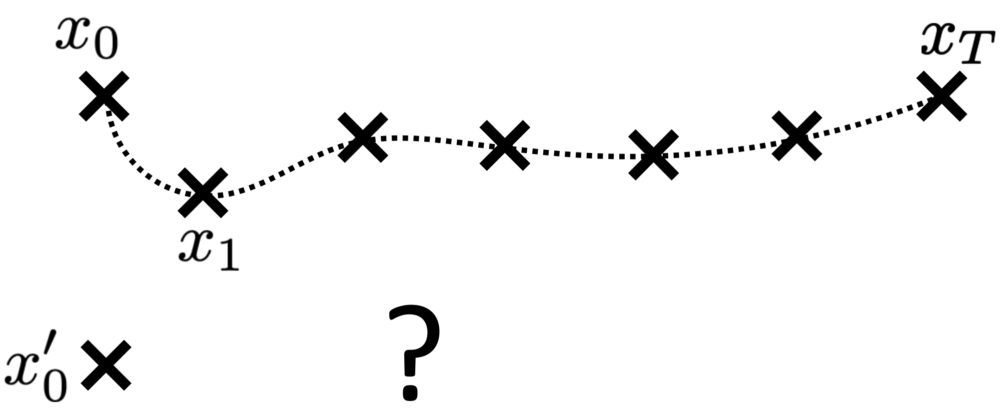


$$\{x\} = x_0, \cdots, x_T$$

$$\{u\} = u_0, \cdots, u_{T-1}$$

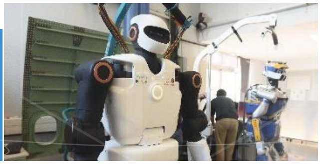


Optimal control problem

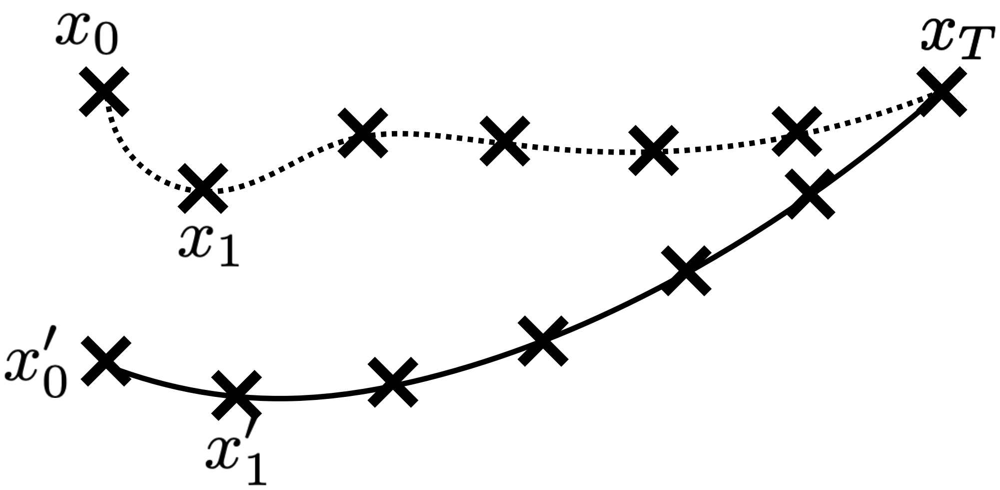


$$\{x\} = x_0, \cdots, x_T$$

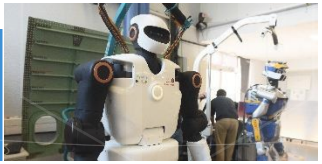
$$\{u\} = u_0, \cdots, u_{T-1}$$



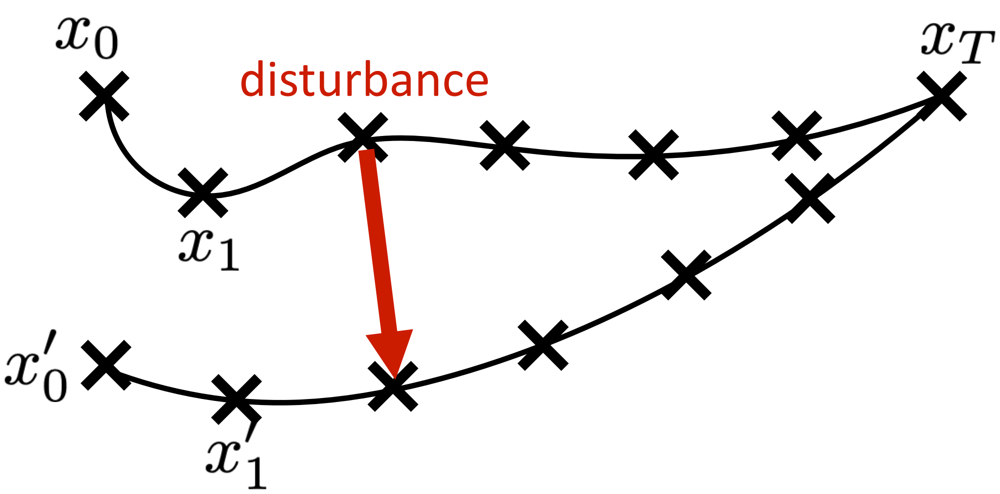
Optimal control problem



$$\{x'\} = x_0', \cdots, x'_T$$
$$\{u'\} = u'_0, \cdots, u'_{T-1}$$



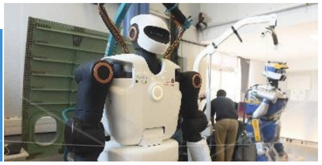
Optimal control problem



$\pi(x)$ => control policy

$\{u\}^*$ the optimal control trajectory

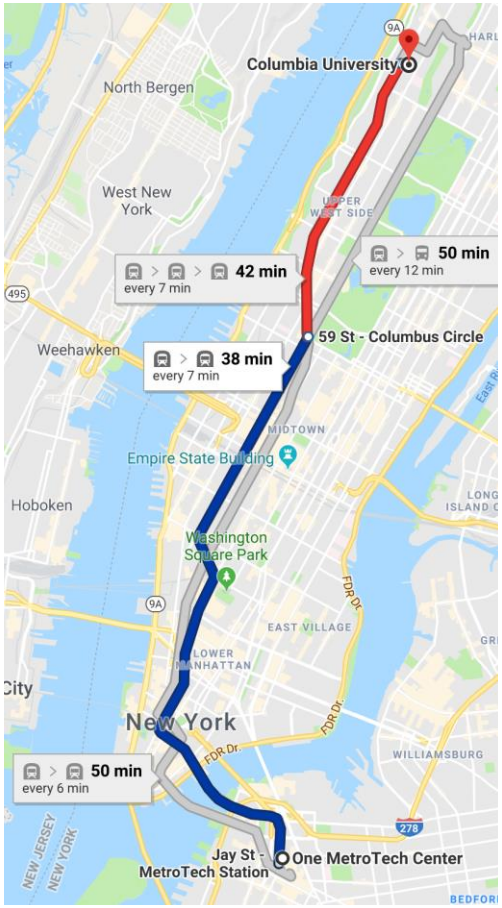
$\pi^*(x)$ the optimal control policy



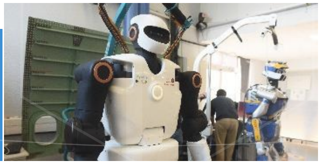
Principle of Optimality

How can we find the optimal control?

The Principle of Optimality breaks down the problem



Subpath of optimal paths are also optimal for then own subproblem



Principle of Optimality

How can we find the optimal control?

The Principle of Optimality breaks down the problem

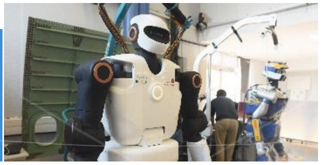
Optimal Cost to Go or Value Function

$$V_t(x_t) = \min_{u_t, \cdots, u_{N-1}} \sum_{k=t}^{T-1} l_k(x_k, u_k) + l_T(x_T)$$

Bellman's Principle of Optimality

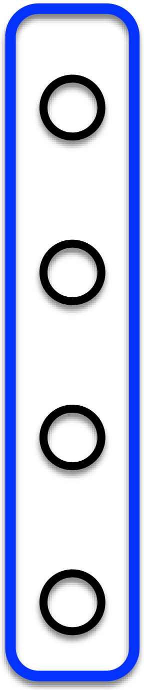
$$V_t(x_t) = \min_{u_t} l_t(x_t, u_t) + V_{t+1}(x_{t+1})$$

$$x_{t+1} = f_t(x_t, u_t)$$

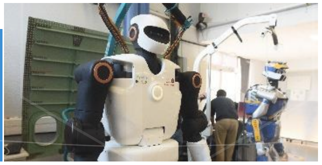


Dynamic Programming

$$V_t(x_t) = \min_{u_t} l_t(x_t, u_t) + V_{t+1}(x_{t+1})$$

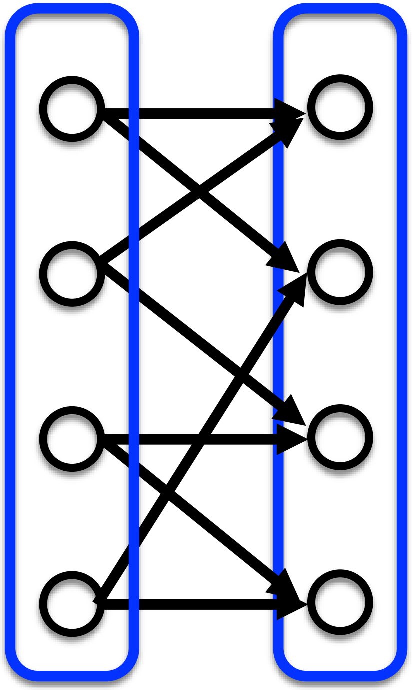


Final States
Stage T
$V_T(x_T)$



Dynamic Programming

$$V_t(x_t) = \min_{u_t} l_t(x_t, u_t) + V_{t+1}(x_{t+1})$$



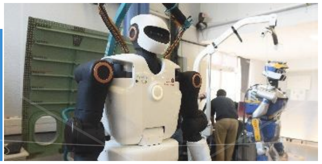
Stage T-1

$V_{T-1}(x_{T-1})$

$\pi_{T-1}(x_{T-1})$

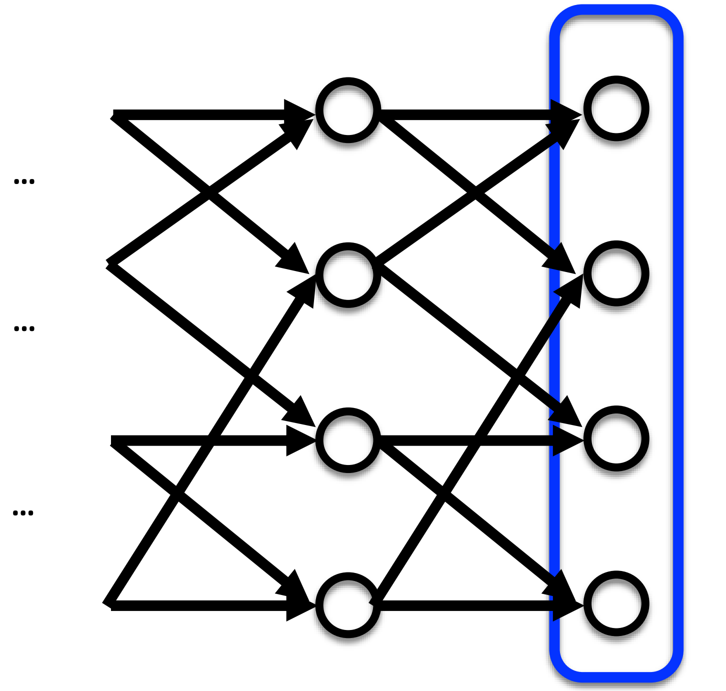
Final States
Stage T

$V_T(x_T)$



Dynamic Programming

$$V_t(x_t) = \min_{u_t} l_t(x_t, u_t) + V_{t+1}(x_{t+1})$$



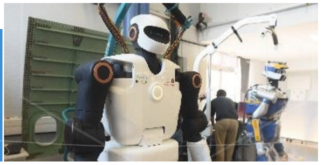
Stage T-1

$V_{T-1}(x_{T-1})$

$\pi_{T-1}(x_{T-1})$

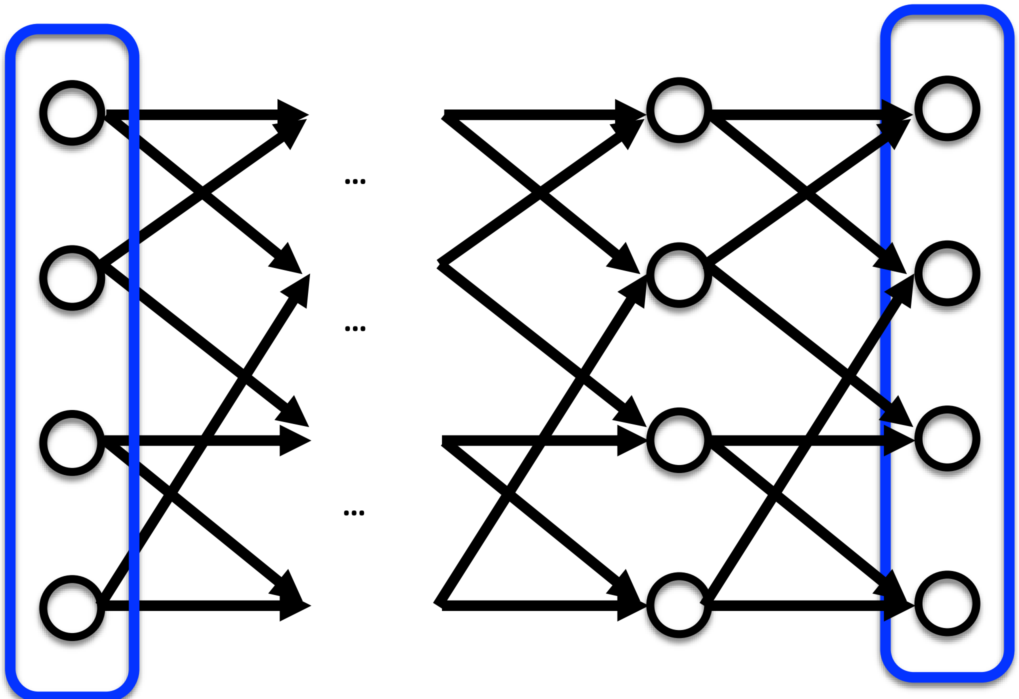
Final States
Stage T

$V_T(x_T)$



Dynamic Programming

$$V_t(x_t) = \min_{u_t} l_t(x_t, u_t) + V_{t+1}(x_{t+1})$$



Stage 0

$V_0(x_0)$

$\pi_0(x_0)$

Stage T-1

$V_{T-1}(x_{T-1})$

$\pi_{T-1}(x_{T-1})$

Final States

Stage T

$V_T(x_T)$

Linear Quadratic Problems

Problems with linear dynamics and quadratic costs can be solved explicitly!

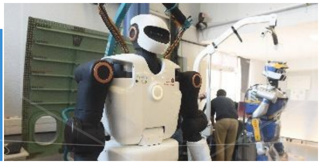
$$x_{t+1} = F_x x_t + F_u u_t$$

Linear dynamics

$$\min \sum_{t=0}^{T-1} (x_t^T L_x x_t + u_t^T L_u u_t) + x_T^T L_x x_T$$

Quadratic cost

$$L_x \geq 0 \quad L_u > 0$$



Linear Quadratic Problems

- Set $W_T = L_x$

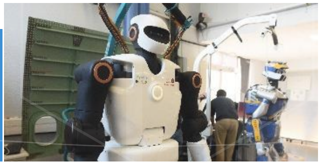
- For t from T-1 to 0, do backward recursion

$$K_t = -(Fu^T\, W_{t+1}\, F_u + L_u)^{-1}\, F_u^T\, W_{t+1}\, F_x$$
$$Q_t = L_x + F_x^T W_t + F_x + F_x^T\, W_{t+1}\, F_u\, K_t$$

<u>Discrete-time Riccati equation</u>

  - The cost-to-go at stage t is $V_t(x_t) = x_t^T\, W_t\, x_t$
  - The optimal policy is $\pi^*(x_t) = K_t\, x_t$

The policy is a linear feedback controller with gain K_t



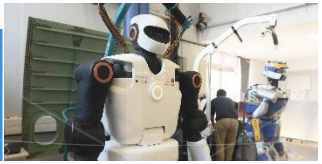
Dynamic Programming

Bellman Equation $V_t(x_t) = \min_{u_t} l_t(x_t, u_t) + V_{t+1}(x_{t+1})$

Problems:

- Curse of dimensionality
- minimization in Bellman equation

$\Rightarrow$ Approximate solution to Bellman equation
(DDP, trajectory optimization, reinforcement learning, etc)



Solving Bellman's Equations

[1] Bonnali'19 ArX:1903.00155
[2] Mordach'14 DOI:2185520.2185539
[3] Posa'14 DOI:0278364913506757
[4] Winkler'18 IEEE:2798285
[5] Rajamaki'17 DOI:3099564.3099579

Bellman's Equation $V_t = \min_{u_t} l(x_t, u_t) + V_{t+1}(f(x_t, u_t))$

Resolution Method:
Stochastic - Deterministic

LQR
(exact solution)

Non LQR
(approximate solution)

Indirect Methods
Pontryagin's Maximum Principle
Rockets, Cars (small dimensions)

GuSTO [1]

Direct Methods
(Most popular in robotics)

"local"
Trajectory optimization

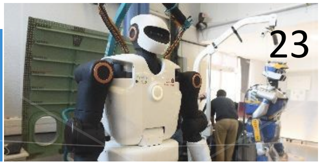
"global"
Value/Policy optimization

Explicit MPC
Q learning
Actor Critic
DDPG, TRPO, PPO

Guided policy
search

Collocation
CIO [2] TOWR [4]
TrajOpt
"Direct" trajectory optim [3]

Shooting
DDP
Multiple shooting
CMAES, PI^2



Should we collocate or shoot?

TRANSCRIPTION

Transcribing:
“representing“ the reality

$$\min_{\substack{\underline{x}:t\longrightarrow x(t)\\ \underline{u}:t\longrightarrow u(t)}} \int_0^T l\big(x(t),u(t)\big)dt + l_T\big(x(T)\big)$$

$$\text{s.t.}\quad \forall t, \dot{x}(t) = f(x(t),u(t))$$

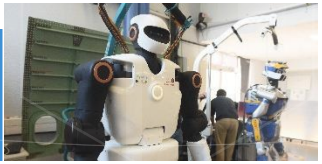
Optimal control problem (OCP)
with continuous variables
(infinite-dimension)

$$\min_{\substack{\underline{x}=\theta_{x1}\ldots\theta_{xn}\\ \underline{u}=\theta_{u1}\ldots\theta_{un}}} \sum_t l\big(x(t|\theta),u(t|\theta)\big) + l_T\big(x(T|\theta)\big)$$

$$\text{s.t.}\quad at\ some\ t, \dot{x}(t|\theta) = f(t|\theta_x,\theta_u)$$

Nonlinear optimization problem (NLP)
with static variables
(finite dimension)

$\theta_x\ \theta_u$ represents the continuous $\underline{x}$,$\underline{u}$ trajectories

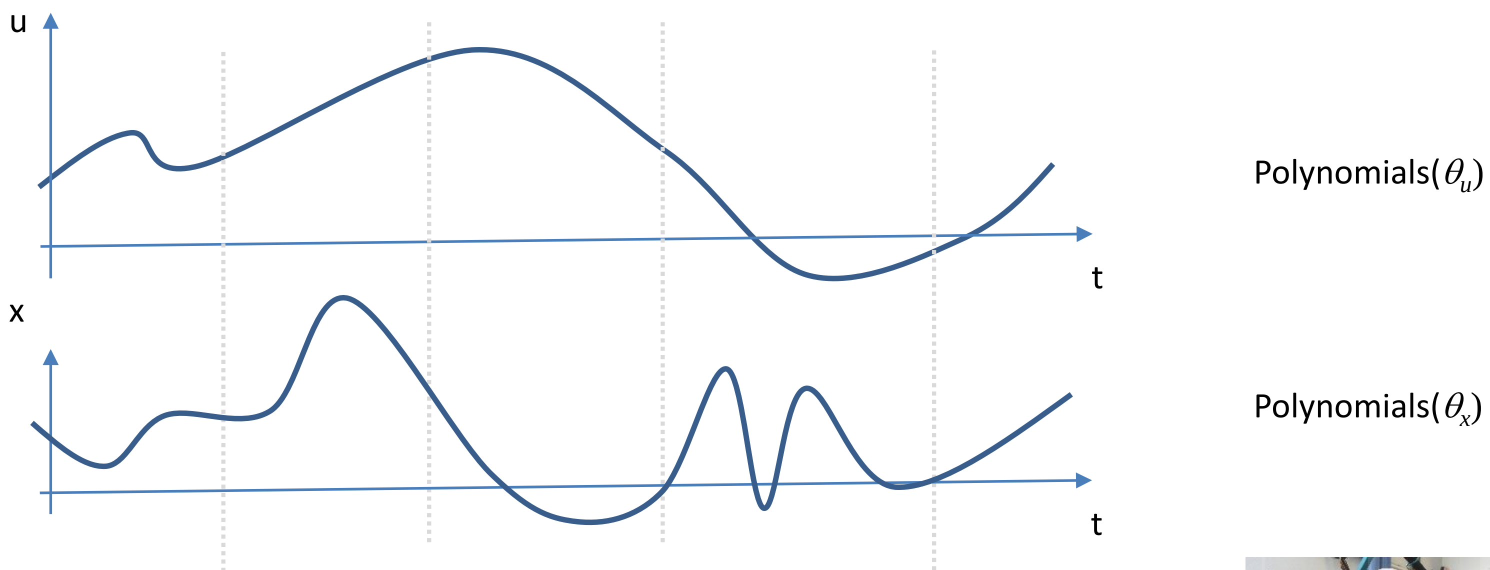


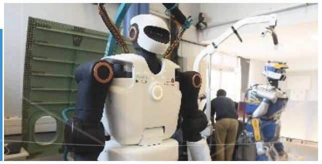
Transcription: shooting versus collocation

$\underline{u}$ is easy to represent (piecewise polynomials)

– what about $\underline{x}$?

- Collocation: $\underline{x}$ is represented by another polynomials





Transcription: shooting versus collocation

$\underline{u}$ is easy to represent (piecewise polynomials)

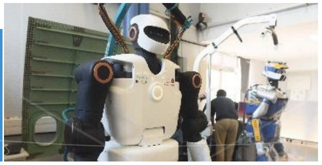
– what about $\underline{x}$?

- Collocation: $\underline{x}$ is represented by another polynomials

Problems:

The solution to $\dot{x}(t) = f(x(t),u(t))$ is not polynomial

The dynamics is only checked at some remote points

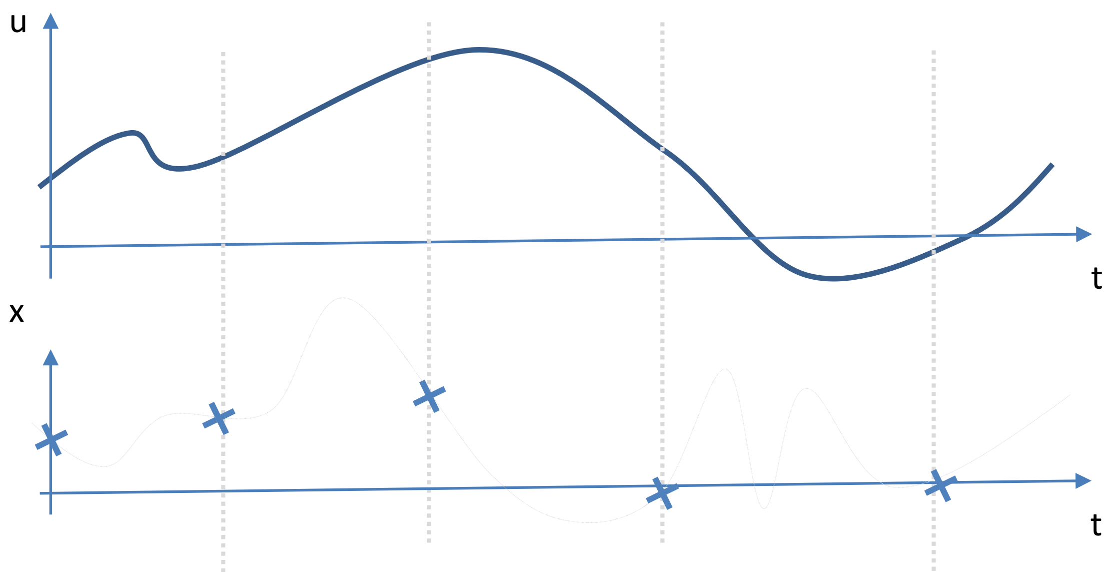


Transcription:
shooting versus collocation

$\underline{u}$ is easy to represent (piecewise polynomials)

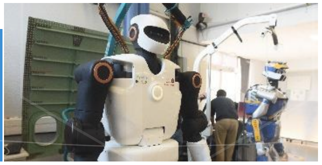
– what about $\underline{x}$?

- Shooting: $\underline{x}$ is represented by and integrator and only evaluated sparsely



Polynomials(θ_u)

$\theta_x = (x_1, \dots x_T)$



Transcription:
shooting versus collocation

$\underline{u}$ is easy to represent (piecewise polynomials)

– what about $\underline{x}$?

- Shooting: $\underline{x}$ is represented by and integrator and only evaluated sparsely

Problems:

The state is sparsely and approximately known

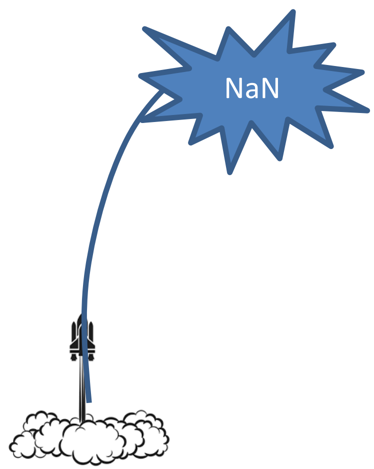
You may need an accurate integrator (complex+costly)

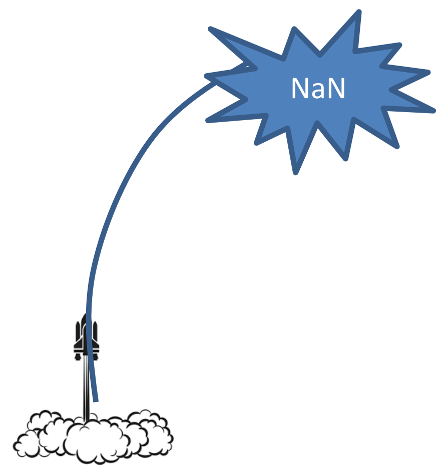
Shooting as control-only problem

$$\min_{\underline{u}=(u_0..u_{T-1})} \sum_t l(x(u_0..u_{t-1}|x_0),u_t) + l_T\left(x(u_0..u_{T-1})\right)$$

where $x(u_0..u_{t-1}|x_0)$ *if a function of* $\underline{u}$

- Unconstrained optimization
- The function $\underline{u}(\underline{x})$ is numerically unstable



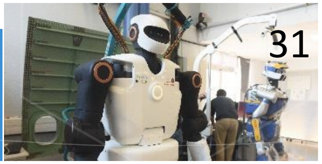


Shooting, pro and cons

- Easy to implement
  - Integrator, derivatives, Newton-descent
- Side effect: you can focus on efficiency

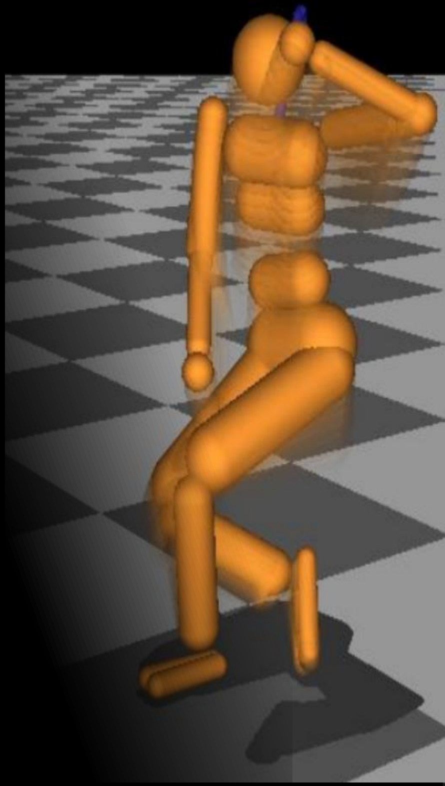
- Numerically unstable
- The initial-guess θ_{xu} should be meaningful

- At then end, maybe we don’t care so much …



Why make your dynamic program *differential*?

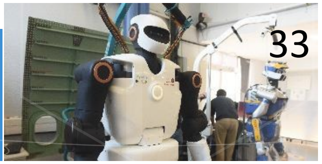
D.D.P.



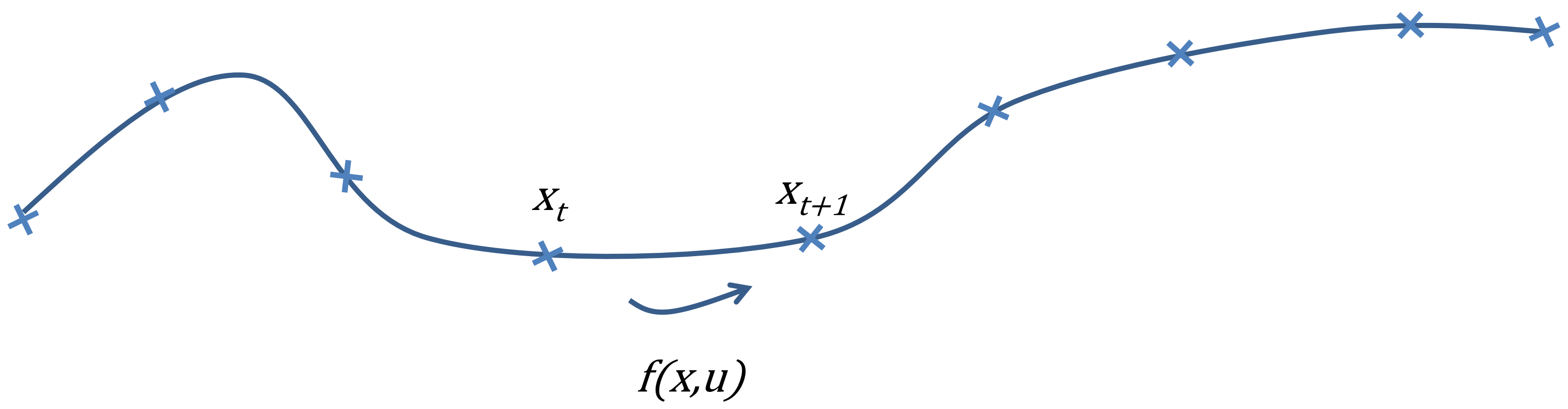
Tassa et al., IROS' 12

Multiple views on DDP

1. DDP as iterative LQR



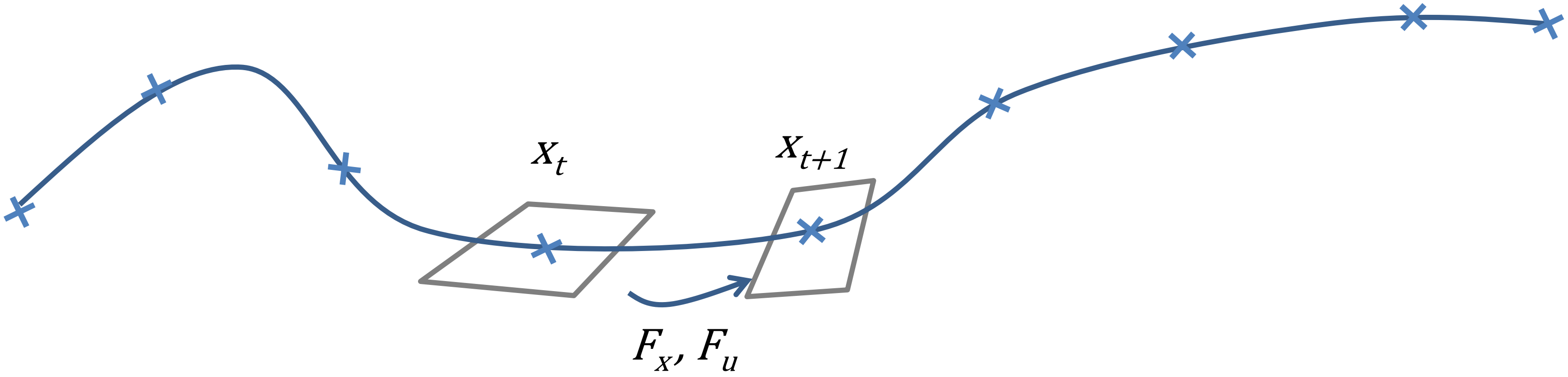
DDP as iterative LQR



- "Next-step" is a nonlinear function

$$\Delta x' = f(x+\Delta x,\ u+\Delta u) - f(x,\ u)$$

DDP as iterative LQR



- "Next-step" is a nonlinear function

$$\Delta x' = f(x+\Delta x, u+\Delta u) - f(x, u)$$

- Approximate by

$$\Delta x' = f(x, u) + F_x \Delta x + F_u \Delta u - f(x, u)$$

DDP as iterative LQR

- Nonlinear optimal control problem

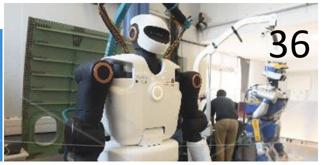
$$\min_{\{x\},\{u\}} \sum_{t=0}^{T-1} l(x_t, u_t) \;+ l_T(x_T)$$

$$\text{s.t.} \quad \forall t=0..T\text{-}1 \quad x_{t+1} = f(x_t, u_t)$$

- Linear-Quadradic problem … solved in Part 1.

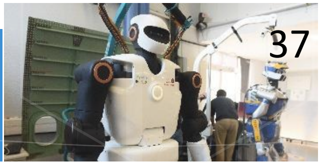
$$\min_{\{\Delta x\},\{\Delta u\}} \sum_{t=0}^{T-1} \begin{pmatrix} L_x \\ L_u \end{pmatrix}^T \begin{pmatrix} \Delta x_t \\ \Delta u_t \end{pmatrix} + \frac{1}{2} \begin{pmatrix} \Delta x_t \\ \Delta u_t \end{pmatrix}^T \begin{pmatrix} L_{xx} & L_{xu} \\ L_{ux} & L_{uu} \end{pmatrix} \begin{pmatrix} \Delta x_t \\ \Delta u_t \end{pmatrix} + \cdots$$

$$\text{s.t.} \quad \forall t=0..T\text{-}1 \; \Delta x_{t+1} = F_x \Delta x_t + F_u \Delta u_t$$



- Algorithm iLQR

```
Initialize with a given trajectory {x0},{u0}
Repeat
      Linearize/Quadratize the OCP
      Compute the LQR policy
      Simulate (roll-out) with LQR regulator
Until local minimum is reached
```



Multiple views on DDP

2. DDP as a 2-pass algorithm

DDP as a 2-pass algorithm

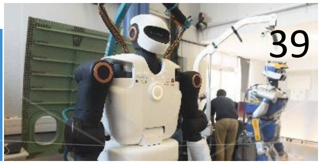
$$V_t = \min_{u_t} l(x_t, u_t) + V_{t+1}(f(x_t, u_t))$$

- Backward propagation

$$Q_t = l(x_t, u_t) + V_{t+1}(f(x_t, u_t))$$

- Greedy optimization

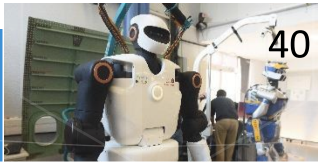
$$V_t = \min_{u_t} Q_t(x_t, u_t)$$



DDP as a 2-pass algorithm

$$Q = l + V'$$

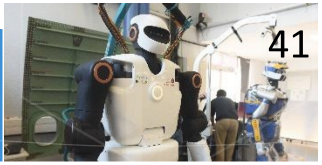
$$V = \min_{u} Q$$



DDP as a 2-pass algorithm

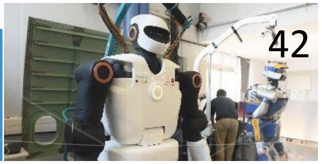
- Pass 1: back-propagate an approximation of V
  - We can solve Belman for quadratic cost and linear dynamics

- Pass 2: forward propagate gains and trajectory



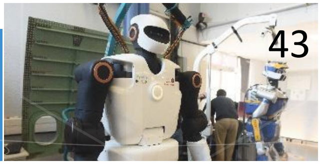
DDP as a 2-pass algorithm

- Pass 1: backpropagate an approximation of V



DDP as a 2-pass algorithm

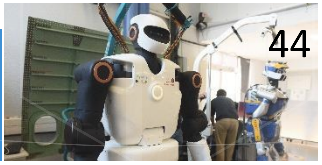
- Pass 2: forward propagate gains and trajectory



DDP as a 2-pass algorithm

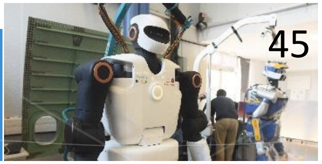
- Globalization (because nonconvexity)
- Line search
  - $u = u^* + k + K\,(x-x^*)$
  - $x' = f(x,u)$

- Regularization
  - $Q_{uu} = L_{uu} + F_u^T\, V_{xx}\, F_u$
  - $k = Q_{uu}^{-1}\, Q_u$
  - $K = Q_{uu}^{-1}\, Q_{ux}$



Multiple views on DDP

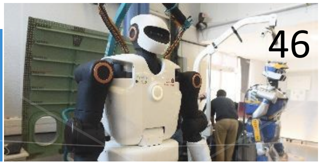
3. DDP as sparse SQP



DDP as sparse SQP

$$\min_{\{x\},\{u\}} \sum_{t=0}^{T-1} l(x_t, u_t) + l_T(x_T)$$

$$\text{s.t.} \quad \forall\ t = 0..T\text{-}1 \quad x_{t+1} = f(x_t, u_t)$$



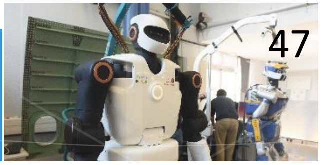
DDP as iterative LQR

- Reminder
- Non linear problem

$$\min_{y} l(y)$$
$$\text{s.t.}\quad f(y)=0$$

- Resulting “linearization”

$$\min_{\Delta y} l(y) + L_y \Delta y + \frac{1}{2}\Delta y^T L_{yy} \Delta y$$
$$\text{s.t.}\quad f(y) + F_y\, \Delta y = 0$$



DDP as sparse SQP

$$\min_{\Delta y} l(y) + L_y \Delta y + \frac{1}{2} \Delta y^T L_{yy} \Delta y$$

$$\text{s.t.} \quad f(y) + F_y \Delta y = 0$$

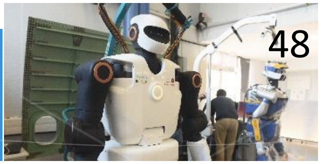
- Lagrangian on the NLP

$$\mathfrak{L}(y, \lambda) = l(y) + \lambda^T f(y)$$

lagrangian

Primal variable

Dual variable (multipliers)



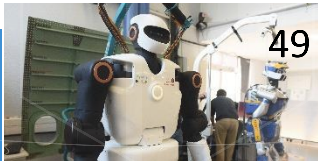
DDP as sparse SQP

$$\min_{\Delta y} l(y) + L_y \Delta y + \frac{1}{2} \Delta y^T L_{yy} \Delta y$$

$$\text{s.t.} \quad f(y) + F_y \Delta y = 0$$

- Lagrangian on the QP

$$\mathfrak{L}(\Delta y, \lambda) = L_y \Delta y + \frac{1}{2} \Delta y^T L_{yy} \Delta y$$
$$+ \lambda^T (F_y \Delta y - f(y))$$



DDP as sparse SQP

$$\min_{\Delta y} l(y) + L_y \Delta y + \frac{1}{2} \Delta y^T L_{yy} \Delta y$$

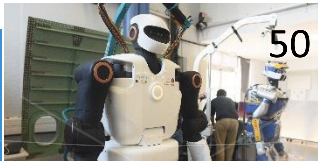
$$\text{s.t.} \quad f(y) + F_y \Delta y = 0$$

- Lagrangian on the QP

$$\mathfrak{L}(\Delta y, \lambda) = L_y \Delta y + \frac{1}{2} \Delta y^T L_{yy} \Delta y + \lambda^T (F_y \Delta y - f(y))$$

- Newton step

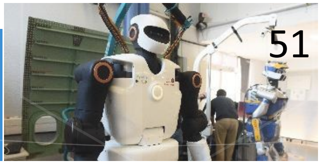
$$\begin{pmatrix} L_{yy} & F_y{}^T \\ F_y & 0 \end{pmatrix} \begin{pmatrix} \Delta y \\ \lambda \end{pmatrix} = \begin{pmatrix} -L_y \\ -f(y) \end{pmatrix}$$



$$\min_{\{\Delta x\},\{\Delta u\}} \sum_{t=0}^{T-1} \begin{pmatrix} L_x \\ L_u \end{pmatrix}^T \begin{pmatrix} \Delta x_t \\ \Delta u_t \end{pmatrix} + \frac{1}{2} \begin{pmatrix} \Delta x_t \\ \Delta u_t \end{pmatrix}^T \begin{pmatrix} L_{xx} & L_{xu} \\ L_{ux} & L_{uu} \end{pmatrix} \begin{pmatrix} \Delta x_t \\ \Delta u_t \end{pmatrix} + \cdots$$

s.t. $\forall$t=0..T-1 $\Delta x_{t+1} = F_x \Delta x_t + F_u \Delta u_t + f_t$

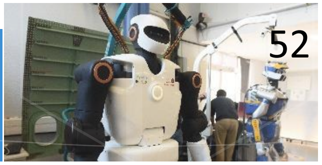
$$\begin{pmatrix} L_{yy} & F_y{}^T \\ F_y & 0 \end{pmatrix} \begin{pmatrix} \Delta y \\ \lambda \end{pmatrix} = \begin{pmatrix} -L_y \\ -f(y) \end{pmatrix}$$



$$\min_{\{\Delta x\},\{\Delta u\}} \sum_{t=0}^{T-1} \begin{pmatrix} L_x \\ L_u \end{pmatrix}^T \begin{pmatrix} \Delta x_t \\ \Delta u_t \end{pmatrix} + \frac{1}{2} \begin{pmatrix} \Delta x_t \\ \Delta u_t \end{pmatrix}^T \begin{pmatrix} L_{xx} & L_{xu} \\ L_{ux} & L_{uu} \end{pmatrix} \begin{pmatrix} \Delta x_t \\ \Delta u_t \end{pmatrix} + \cdots$$

s.t. $\forall t=0..T\text{-}1 \quad \Delta x_{t+1} = F_x \Delta x_t + F_u \Delta u_t + f_t$

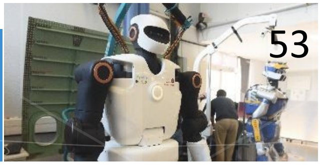
$$\left[\begin{array}{cccc|ccc|ccc}
L_{xx} & & & & L_{xu} & & & -I & F_x^T & \\
& \ddots & & & & \ddots & & \ddots & \ddots & \\
& & L_{xx} & & & & L_{xu} & & -I & F_x^T \\
& & & L_{xx} & & & & & & -I \\
\hline
L_{ux} & & & & L_{uu} & & & F_u^T & & \\
& \ddots & & & & \ddots & & & \ddots & \\
& & L_{ux} & & & & L_{uu} & & & F_u^T \\
\hline
-I & & & & & & & & & \\
F_x & -I & & & F_u & & & & & \\
& \ddots & \ddots & & & \ddots & & & & \\
& & F_x & -I & & & F_u & & &
\end{array}\right]
\left[\begin{array}{c}
\Delta x_0 \\ \vdots \\ \Delta x_{T-1} \\ \Delta x_T \\ \hline \Delta u_0 \\ \vdots \\ \Delta u_{T-1} \\ \hline \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_{T-1}
\end{array}\right]
= -
\left[\begin{array}{c}
L_x \\ \vdots \\ L_x \\ L_x \\ \hline L_u \\ \vdots \\ L_u \\ \hline f_0 \\ f_1 \\ \vdots \\ f_{T-1}
\end{array}\right]$$



$$\min_{\{\Delta x\},\{\Delta u\}} \sum_{t=0}^{T-1} \begin{pmatrix} L_x \\ L_u \end{pmatrix}^T \begin{pmatrix} \Delta x_t \\ \Delta u_t \end{pmatrix} + \frac{1}{2} \begin{pmatrix} \Delta x_t \\ \Delta u_t \end{pmatrix}^T \begin{pmatrix} L_{xx} & L_{xu} \\ L_{ux} & L_{uu} \end{pmatrix} \begin{pmatrix} \Delta x_t \\ \Delta u_t \end{pmatrix} + \cdots$$

s.t. $\forall t=0..T\text{-}1 \quad \Delta x_{t+1} = F_x \Delta x_t + F_u \Delta u_t + f_t$

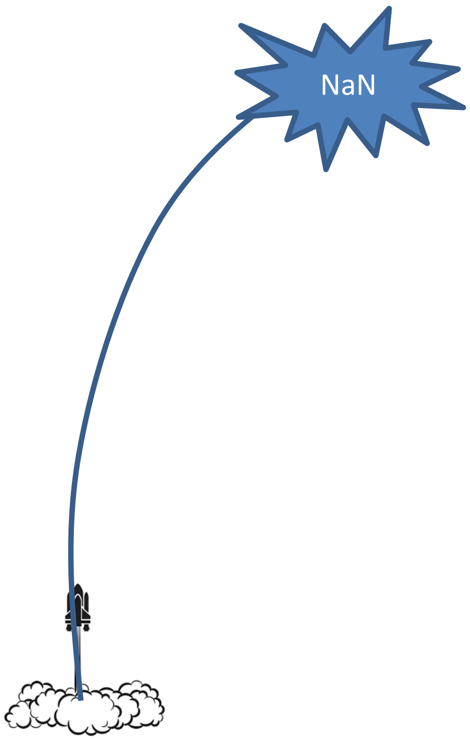
$$\left[\begin{array}{cccc|ccc|cccc}
L_{xx} & & & & L_{xu} & & & -I & F_x^T & & \\
 & \ddots & & & & \ddots & & & \ddots & \ddots & \\
 & & L_{xx} & & & & L_{xu} & & & -I & F_x^T \\
 & & & L_{xx} & & & & & & & -I \\
\hline
L_{ux} & & & & L_{uu} & & & & F_u^T & & \\
 & \ddots & & & & \ddots & & & & \ddots & \\
 & & L_{ux} & & & & L_{uu} & & & & F_u^T \\
\hline
-I & & & & & & & & & & \\
F_x & -I & & & F_u & & & & & & \\
 & \ddots & \ddots & & & \ddots & & & & & \\
 & & F_x & -I & & & F_u & & & &
\end{array}\right]
\begin{bmatrix} \Delta x_0 \\ \vdots \\ \Delta x_{T-1} \\ \Delta x_T \\ \hline \Delta u_0 \\ \vdots \\ \Delta u_{T-1} \\ \hline \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_{T-1} \end{bmatrix}
= -
\begin{bmatrix} L_x \\ \vdots \\ L_x \\ L_x \\ \hline L_u \\ \vdots \\ L_u \\ \hline f_0 \\ f_1 \\ \vdots \\ f_{T-1} \end{bmatrix}$$

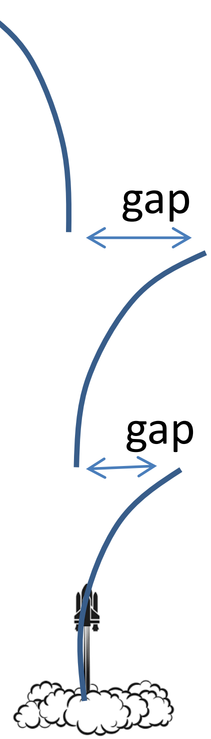


What is Crocoddyl good for, and what is beyond?

BEYOND DDP

Multiple shooting

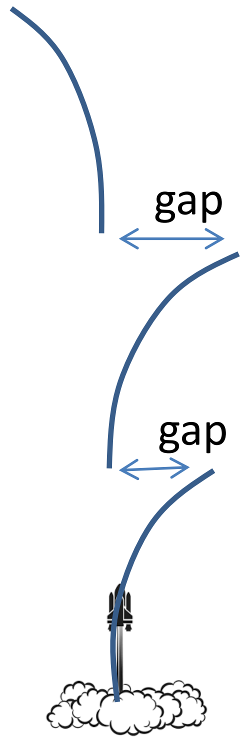


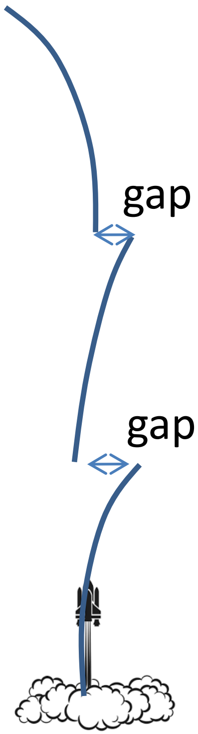


Single shooting
“Your control is bad! “

Multiple shooting
“Your control is bad! “

Multiple shooting

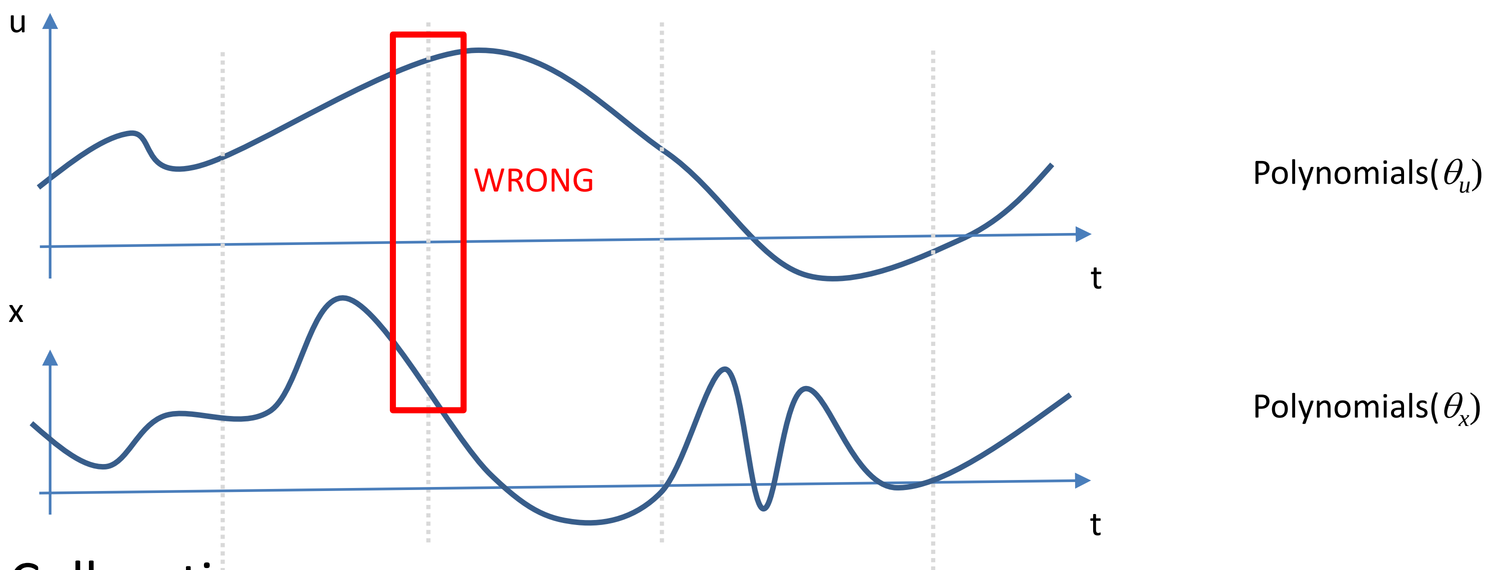




"Your control is bad! "

"Still bad, but better"

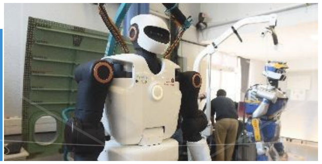
Interpretation of dynamics violation



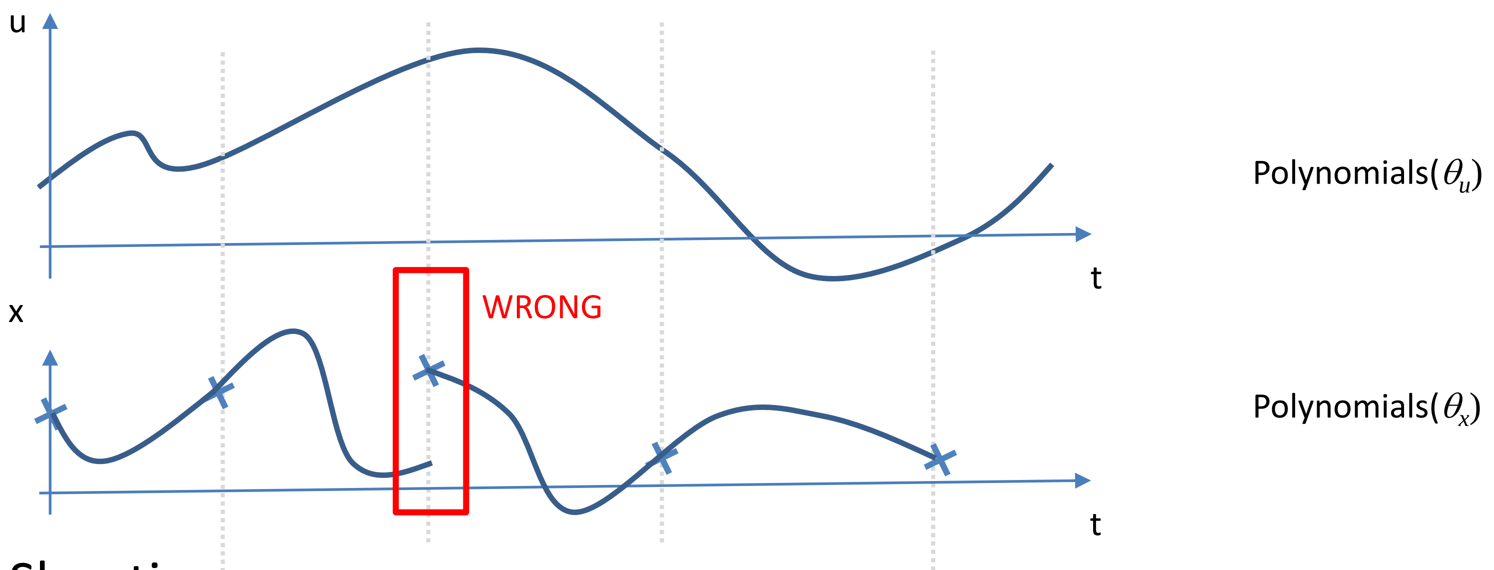
- Collocation:

  We have state and control trajectories

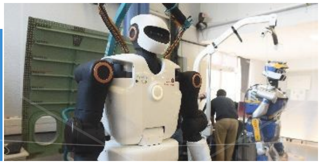
  … and they do not match



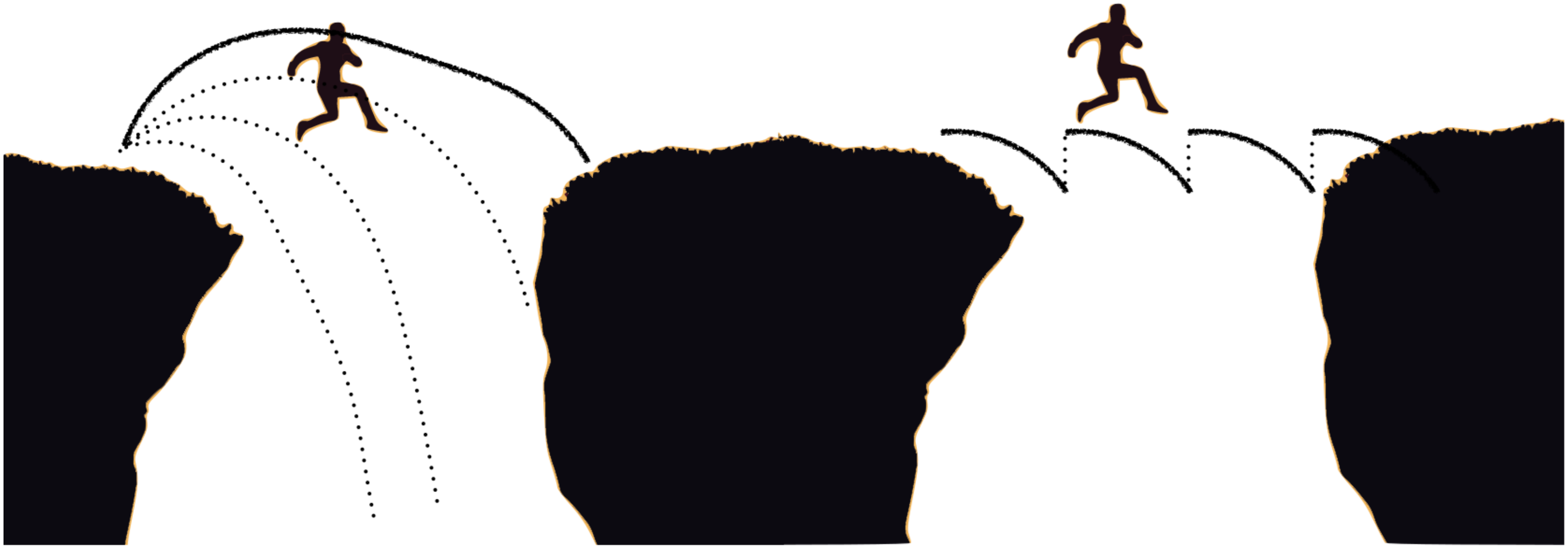
Interpretation of dynamics violation



- Shooting:
  We have a set of state points
  … and the integrator does not reach them



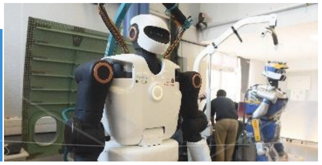
Example of jumping



Thanks Rohan for the illustration

Single

Multiple



Constraints: penalty and projection

$$\min_{\{x\},\{u\}} \sum_{t=0}^{T-1} l(x_t, u_t) \; + l_T(x_T)$$

$$\text{s.t.} \quad \forall t=0..T\text{-}1 \quad x_{t+1} = f(x_t, u_t)$$

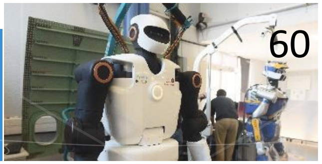
$$\forall t=0..T \;\; g(x_t, u_t) \leq 0$$

By projection

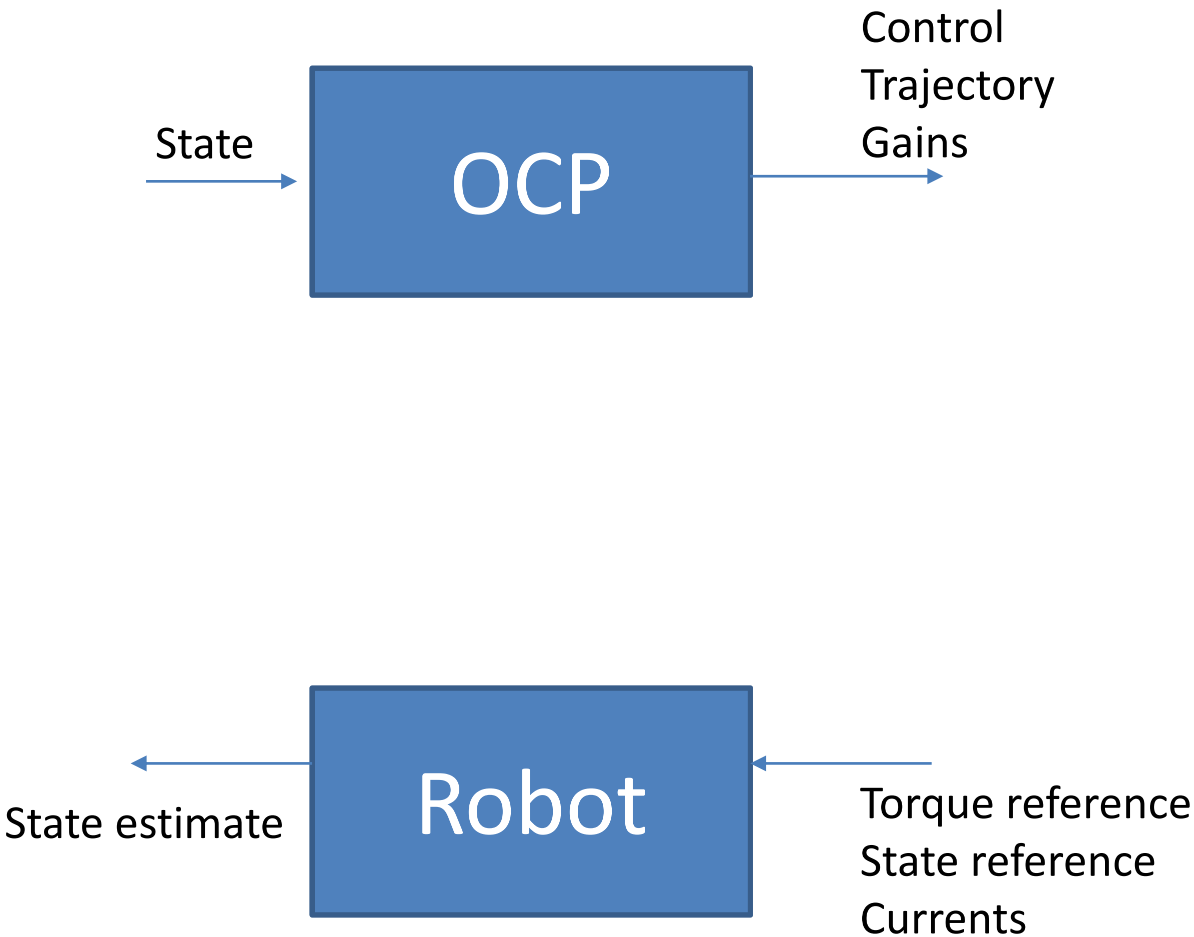
SQP, active set

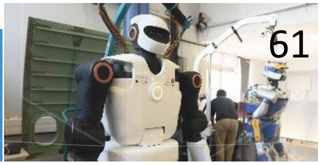
By penalty

Interior point, augmented lagrangian

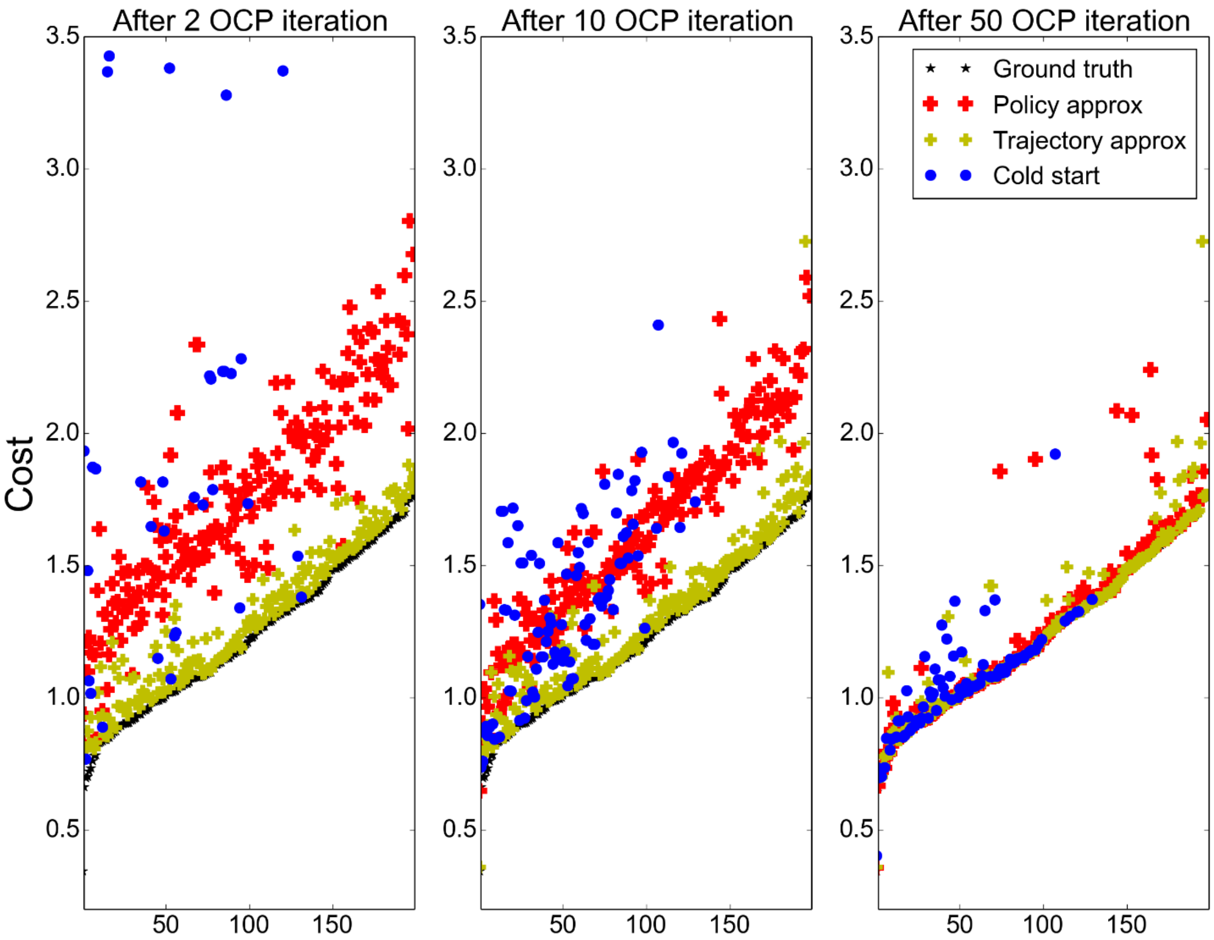


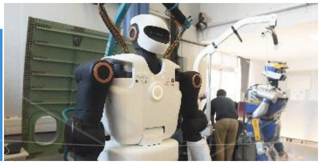
- Closing the loop on the real robot





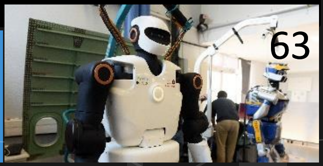
Importance of the warm start





Start to warm-up your fingers

THE END



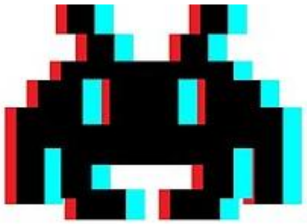
Take-home messages

Numerical problems (few/none discrete constraints)

- nonconvex ... warm start needed
- very constrained ... mostly feasibility problems

The formulation/transcription is our central problem

- expert+math knowledge
- keep generalization



Optimal control = reinforcement learning

- train offline
- generalize online