Agimus Winter School
11/12/2023 - 15/12/2023
Banyuls (France)

AGIMUS
INNOVATIVE ROBOTICS FOR AGILE PRODUCTION

Introduction to optimal control & trajectory optimization

Nicolas Mansard
CNRS

Funded by the European Union under GA no 101070165.

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 are our toolboxes Crocoddyl/Aligator good for, and what is beyond?

What can we do with optimal control?

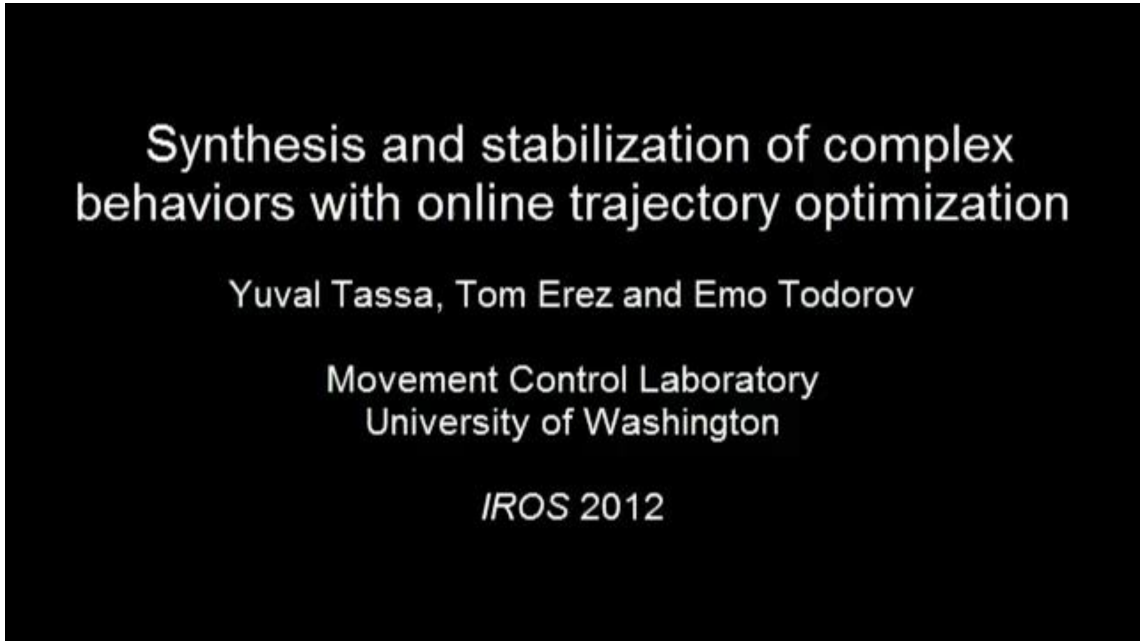
VIDEO INTRODUCTION

Autonomous Driving

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

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]



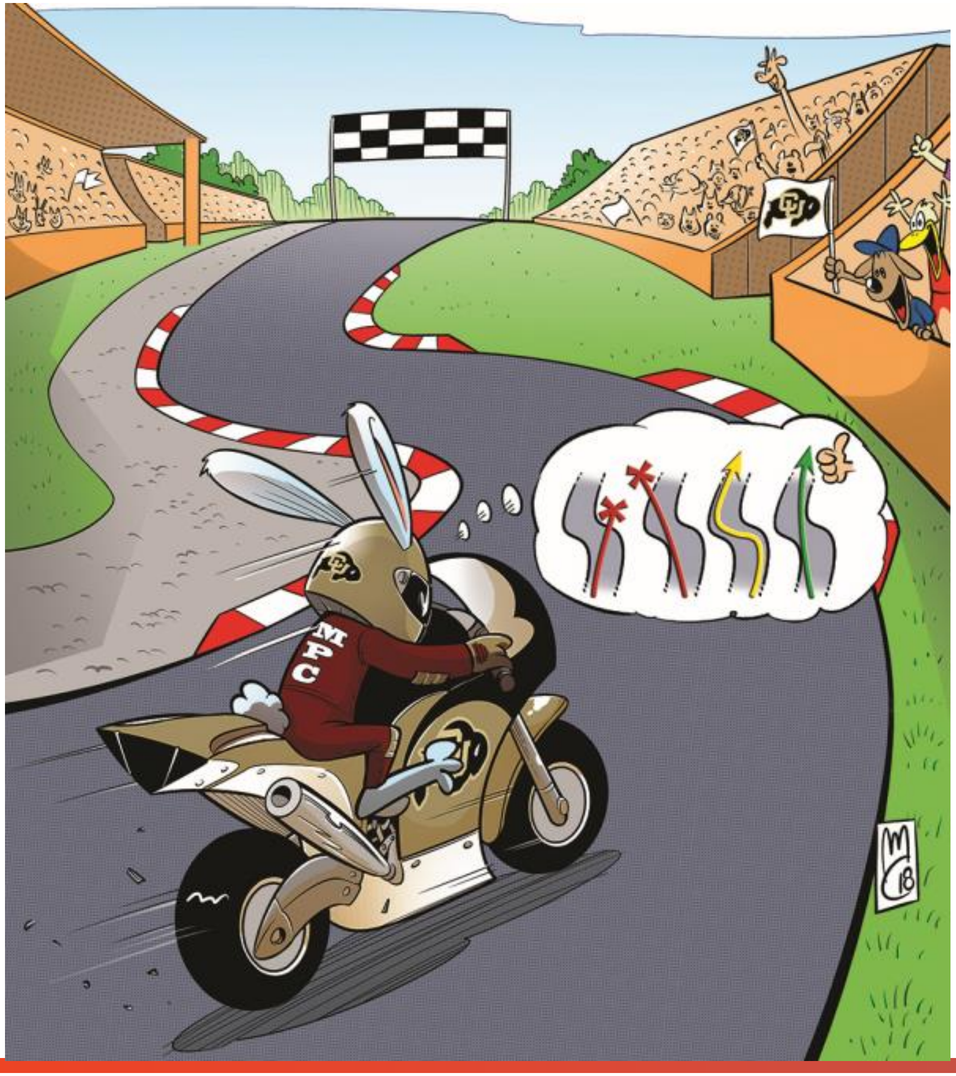
[Tassa et al. 2010]
DDP with Full-Body Dynamics
(realtime control)

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

What is dynamic programing

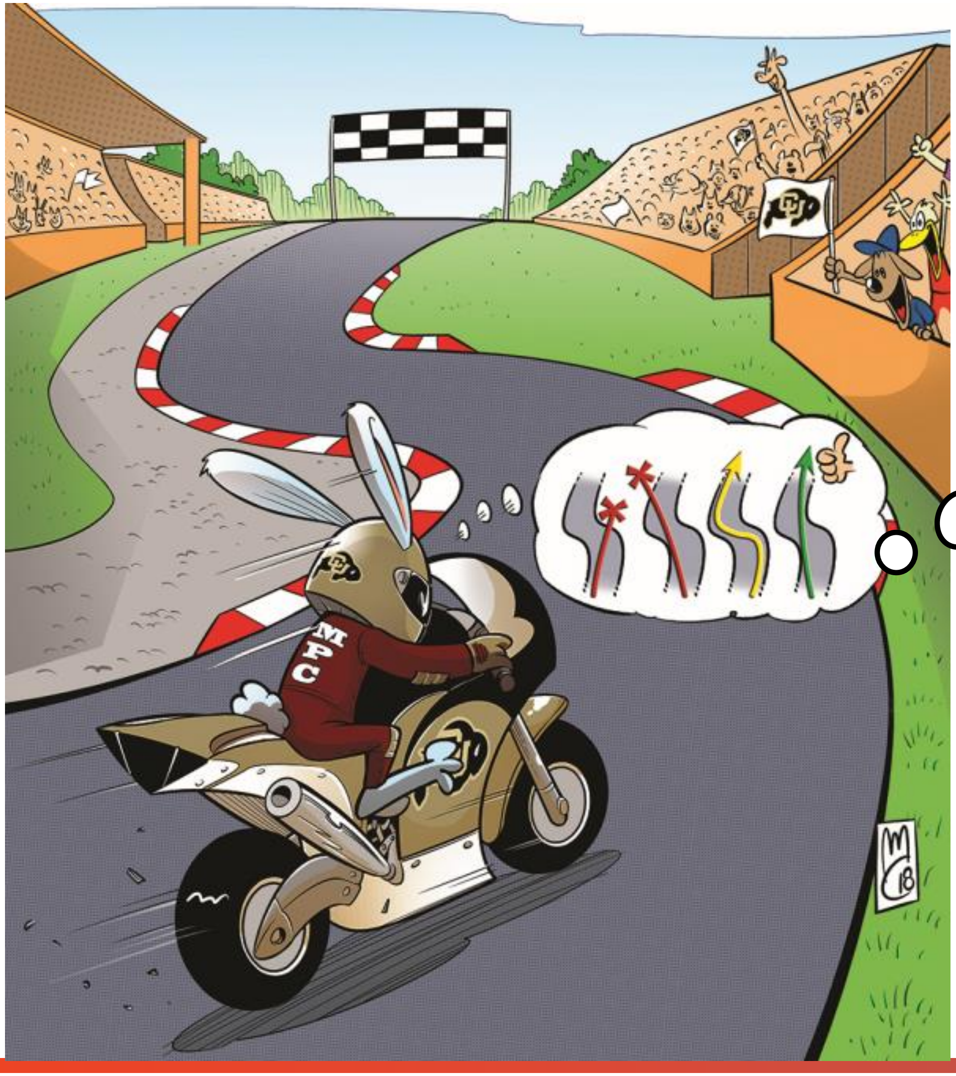
INTRODUCTION
TO BELMAN's EQUATIONS

Starting example



Original artwork by Michele Carminati,
commissioned by Marco M. Nicotra (U. Colorado Boulder)

Starting example



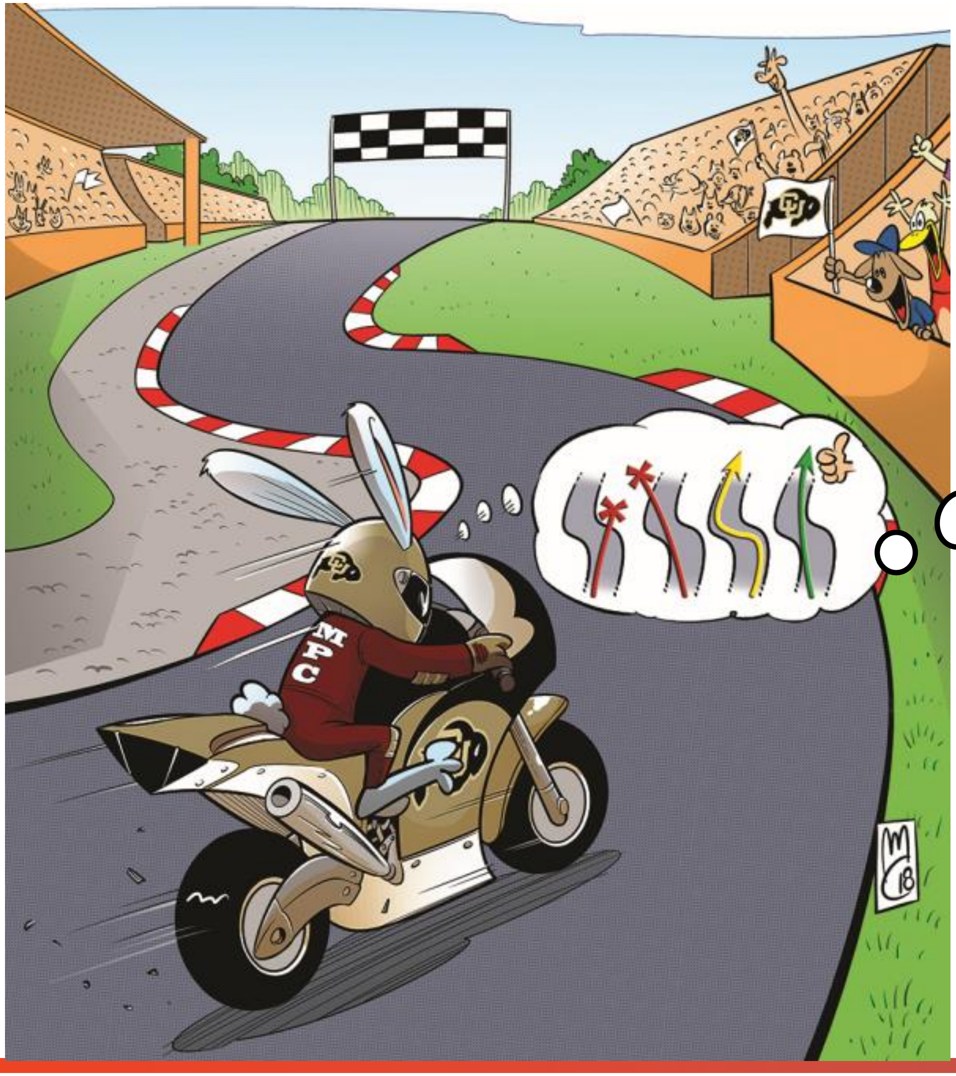
Original artwork by Michele Carminati, commissioned by Marco M. Nicotra (U. Colorado Boulder)

Decide: future robot trajectory

Optimizing: an objective function
(e.g. minum energy)

Satisfying the constraints:
- Known initial state
- Known evolution model
- And others (e.g stay on the road

Starting example



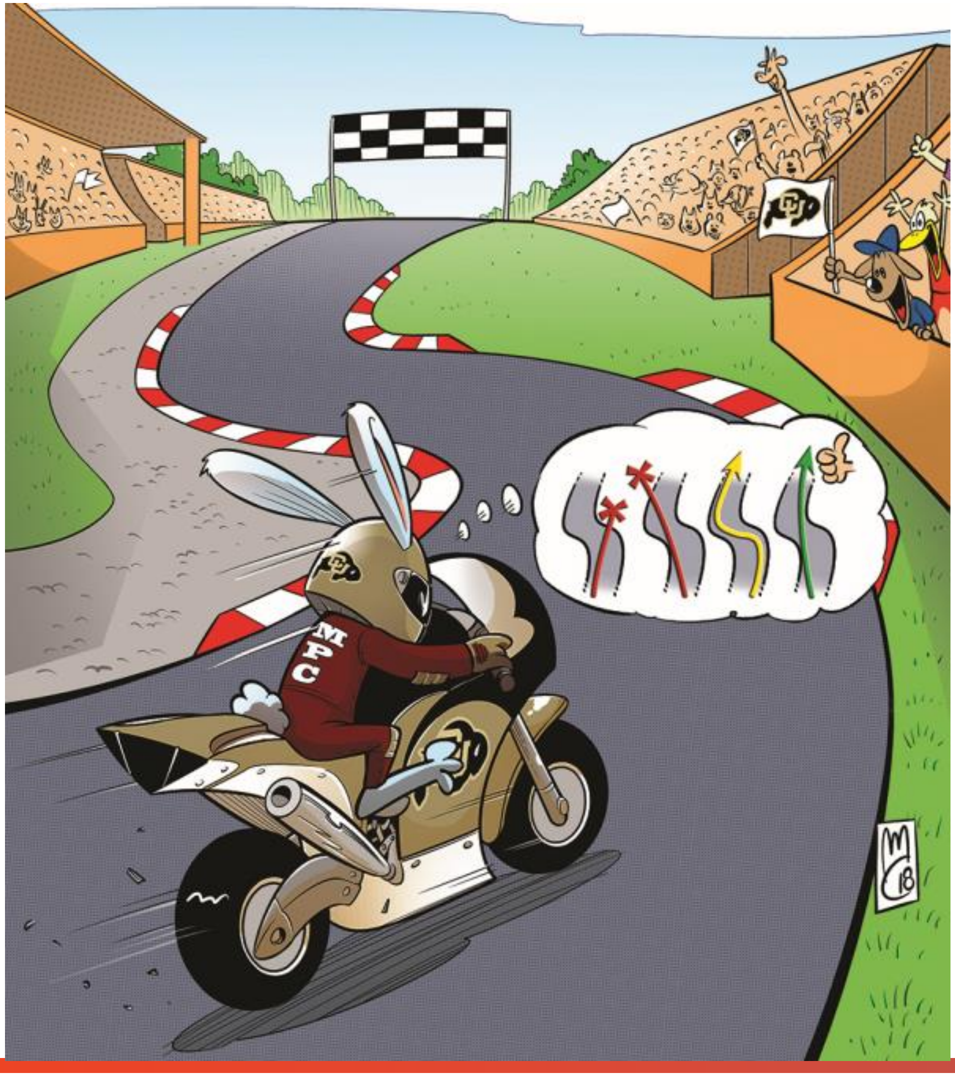
Original artwork by Michele Carminati, commissioned by Marco M. Nicotra (U. Colorado Boulder)

$$\min_{\substack{X=(Q,\dot{Q}),\\ U=\tau}} \int_0^T \sum_l l(x_t, u_t)\, dt$$

so that $\quad x_0 = \hat{x}$

$\qquad\qquad \forall t, \;\; \dot{x}(t) = f(x(t), u(t))$

Starting example



Original artwork by Michele Carminati, commissioned by Marco M. Nicotra (U. Colorado Boulder)

$$\min_{\substack{X=(Q,\dot{Q}),\\ U=\tau}} \int_0^T \sum_l l(x_t, u_t)\, dt$$

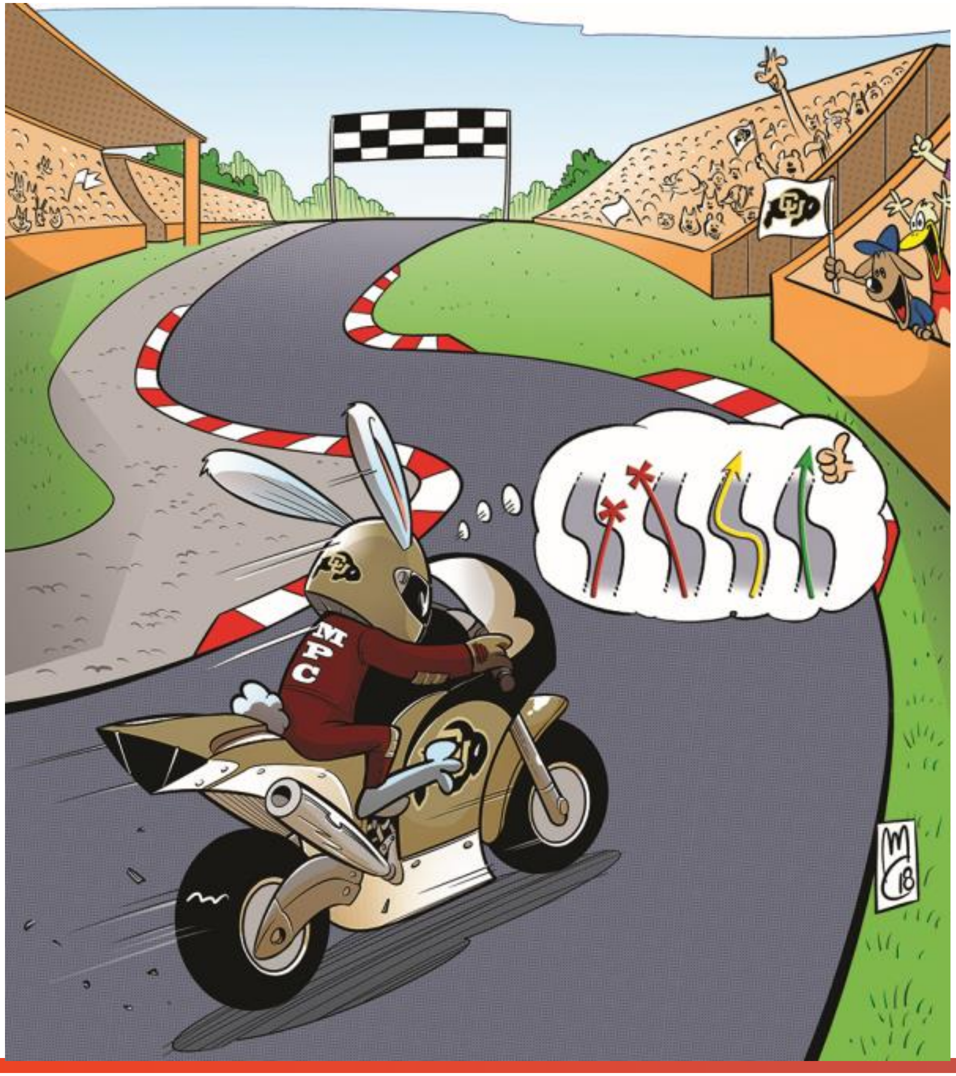
so that $\quad \forall t,\ \dot{x}(t) = f(x(t), u(t))$

$u_0 = \pi(x_0)$

$\{x\},\{u\}$

$\approx +\infty$

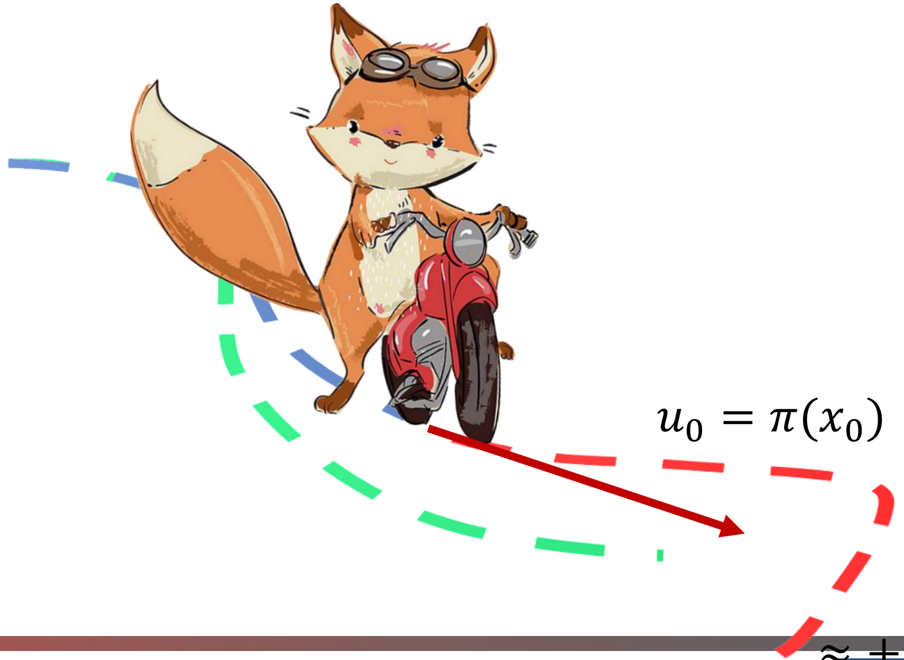
Starting example



Original artwork by Michele Carminati,
commissioned by Marco M. Nicotra (U. Colorado Boulder)

$$\min_{\substack{X=(Q,\dot{Q}),\\ U=\tau}} \int_0^T \sum_l l(x_t, u_t)\, dt$$

so that $\quad \forall t,\ \dot{x}(t) = f(x(t), u(t))$



$u_0 = \pi(x_0)$

$\approx +\infty$

Optimal control problem

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

Find control inputs to minimize cost

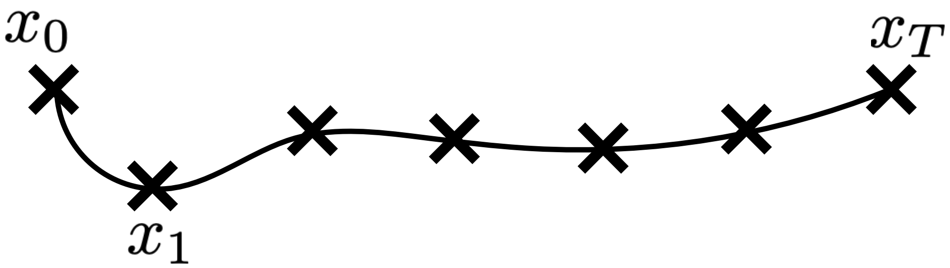
stage costs

terminal cost

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

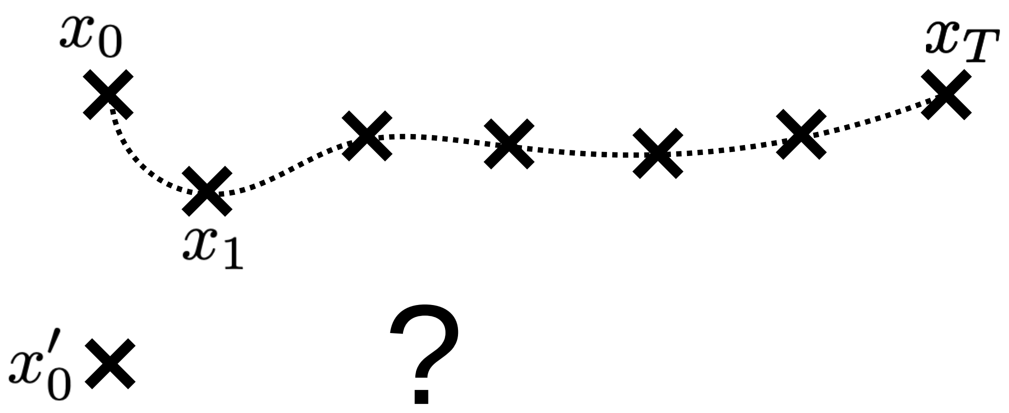
$$g(x_t, u_t) \leq 0$$ state and control constraints

Optimal control problem



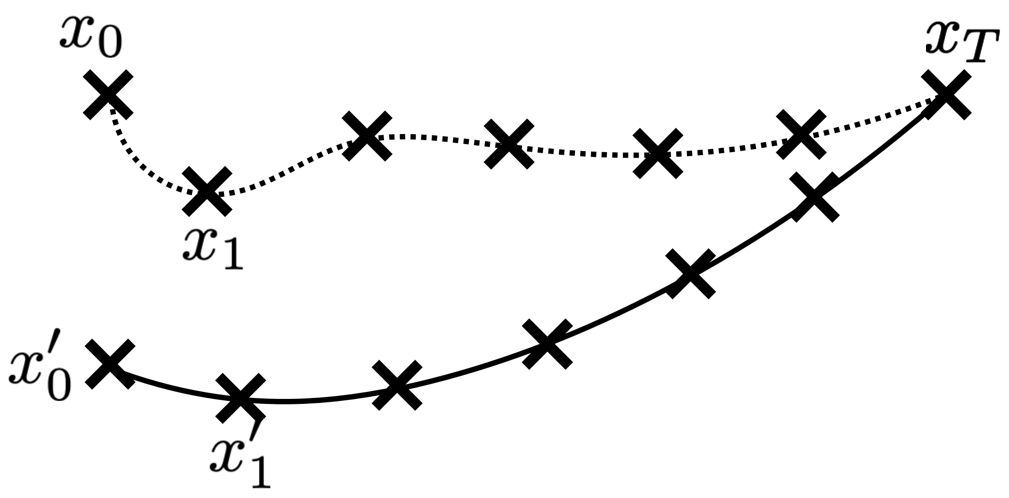
$$\{x\} = x_0, \cdots, x_T$$
$$\{u\} = u_0, \cdots, u_{T-1}$$

Optimal control problem



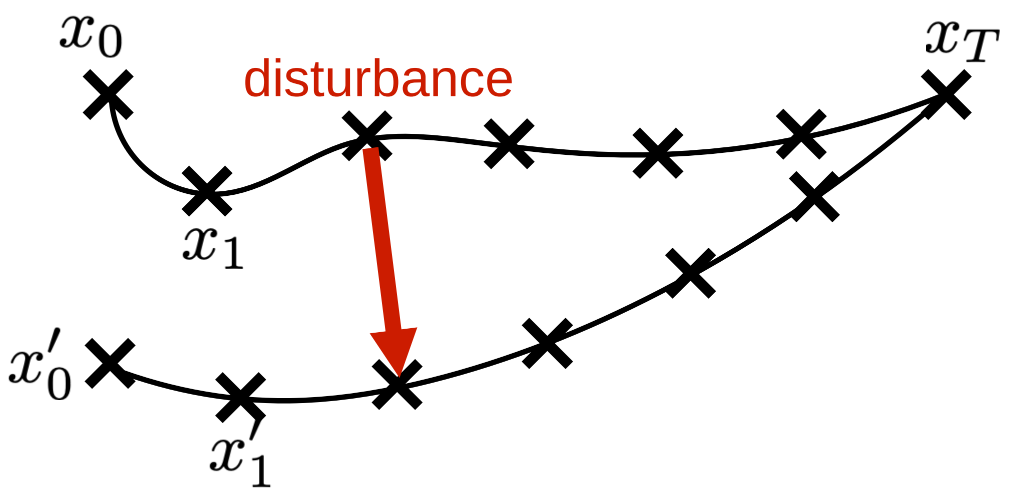
$$\{x\} = x_0, \cdots, x_T$$
$$\{u\} = u_0, \cdots, u_{T-1}$$

Optimal control problem



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

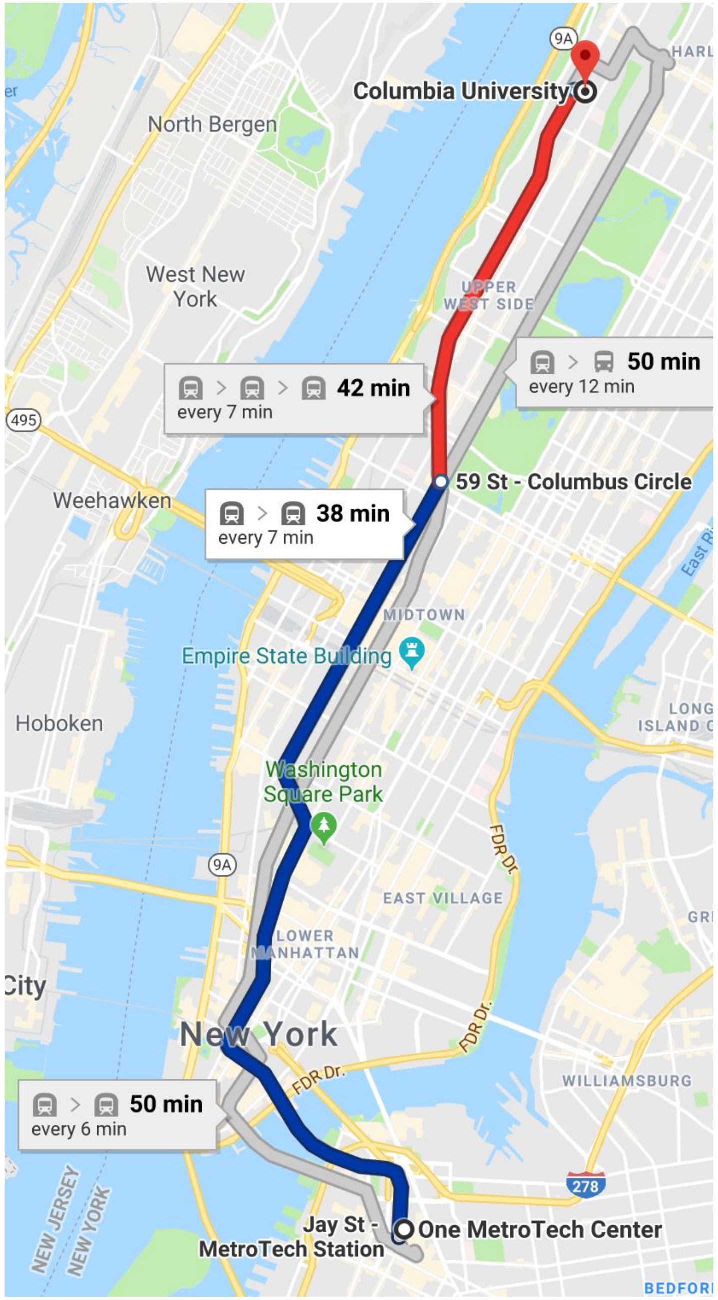
Optimal control problem



$\pi(x)$ => control policy

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

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



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*

How can we find the optimal control?

- The Principle of Optimality breaks down the problem

Optimal Cost to Go or Value Function

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

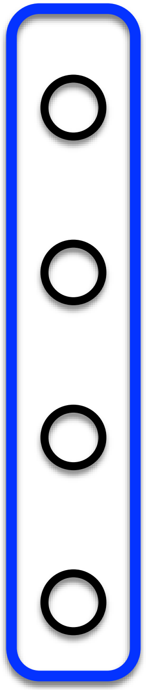
Bellman's Principle of Optimality

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

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

Dynamic programming

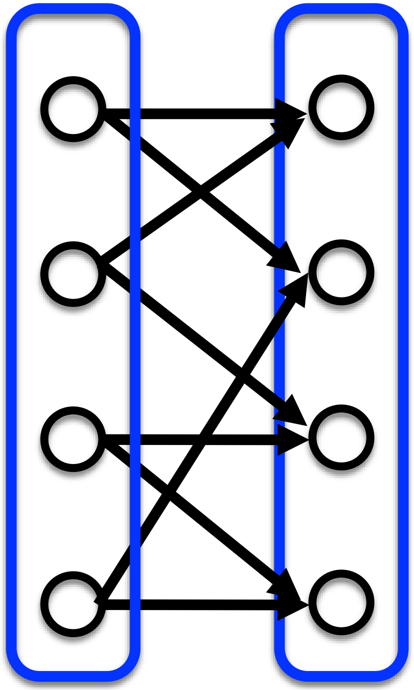
$$V_t(x_t) = \min_{u_t} l_t(x_t, u_t) + V_{t+1}(x_{t+1})$$



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

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

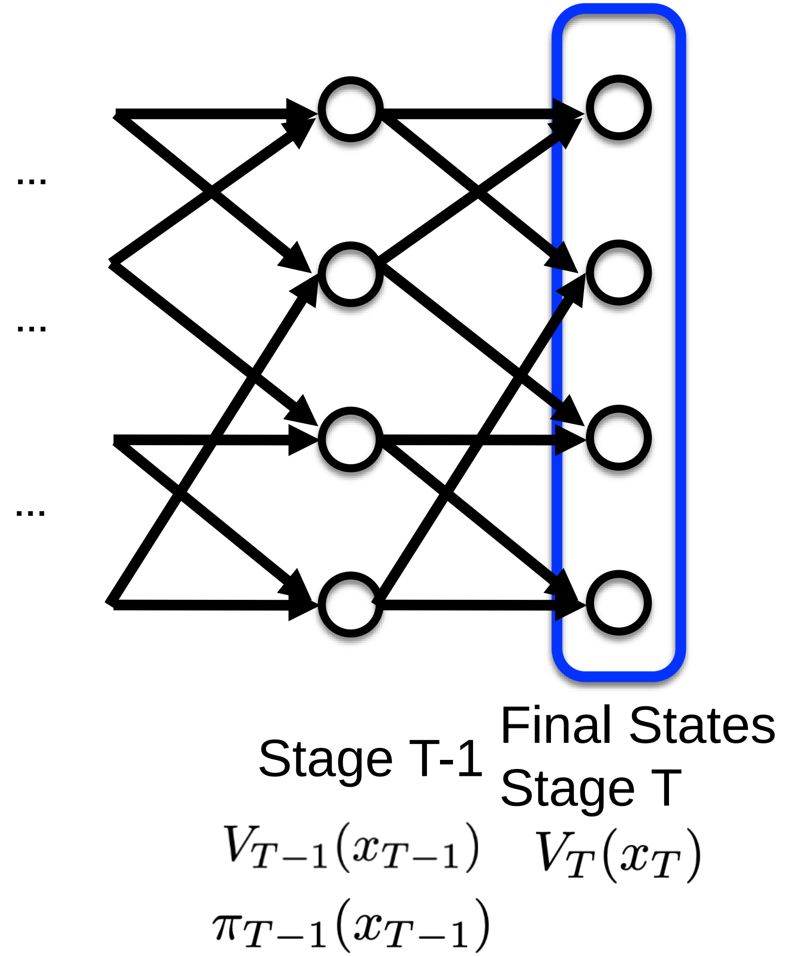
$\pi_{T-1}(x_{T-1})$

Final States
Stage T

$V_T(x_T)$

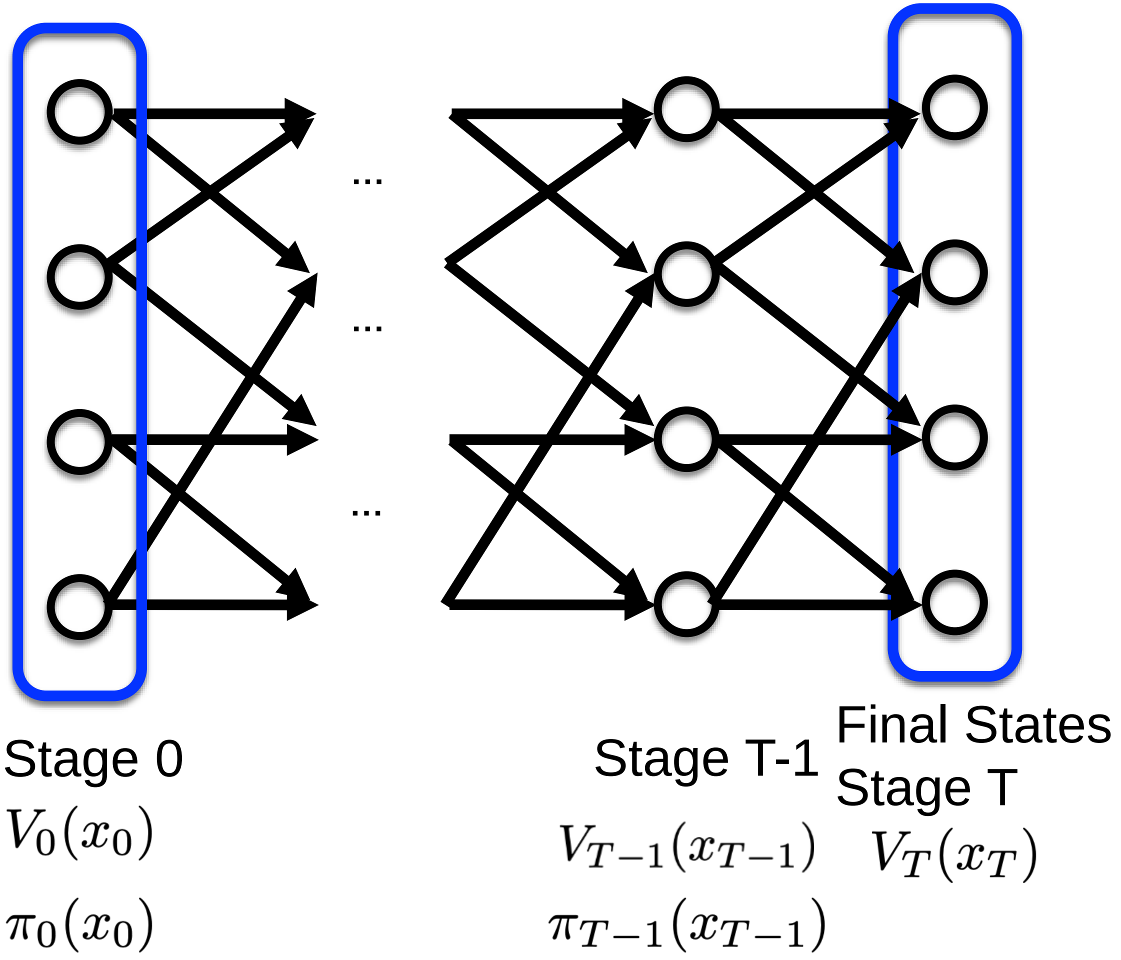
Dynamic programming

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



Dynamic programming

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



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)

[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

Solving Bellman's Equations

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

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

Shooting
DDP
Multiple shooting
CMAES, PI^2

Guided
policy
search

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)

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

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

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

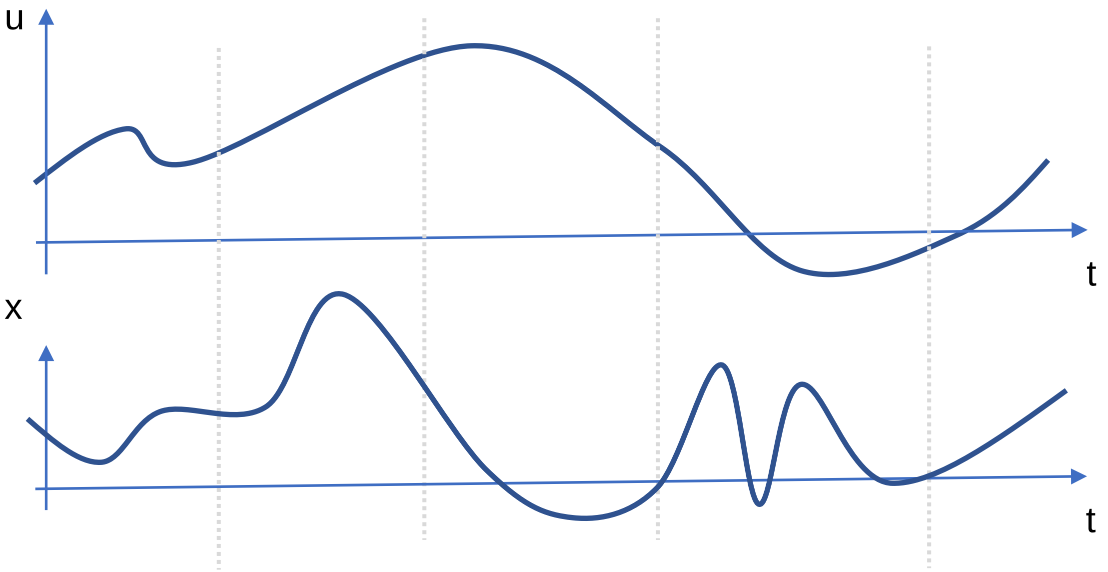
$\theta_x\ \theta_u$ represents the continuous $\underline{x}$,$\underline{u}$ trajectories

Transcribing: *“representing“ the reality*

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

– what about $\underline{x}$?

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



Polynomials(θ_u)

Polynomials(θ_x)

Transcribing: *“representing“ the reality*

$\underline{u}$ 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

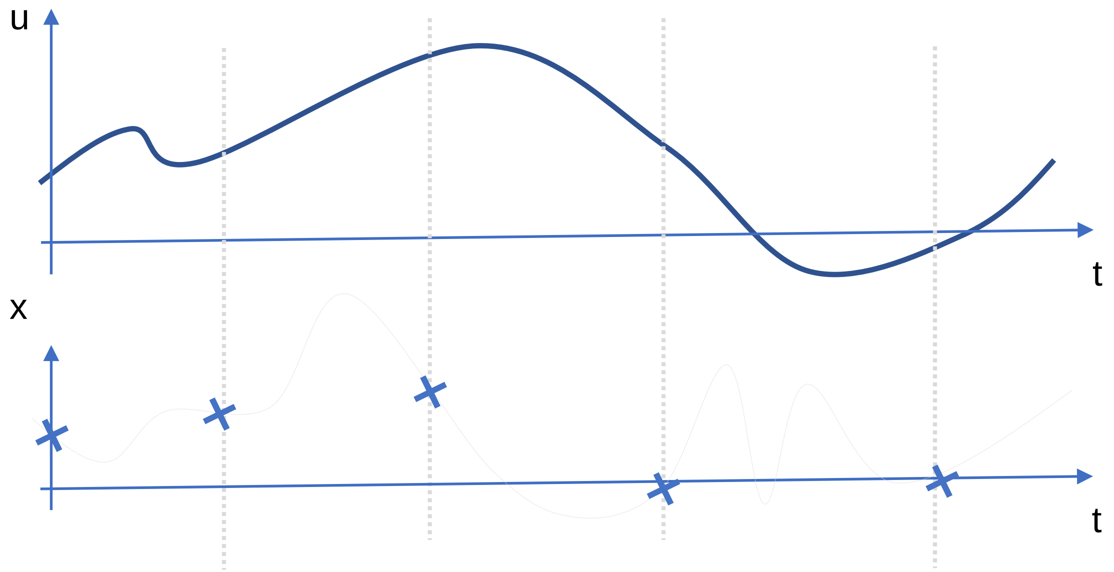
The dynamics is only checked at some remote points

Transcribing: *“representing“ the reality*

$\underline{u}$ 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)

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

Transcribing: *"representing" the reality*

$\underline{u}$ 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

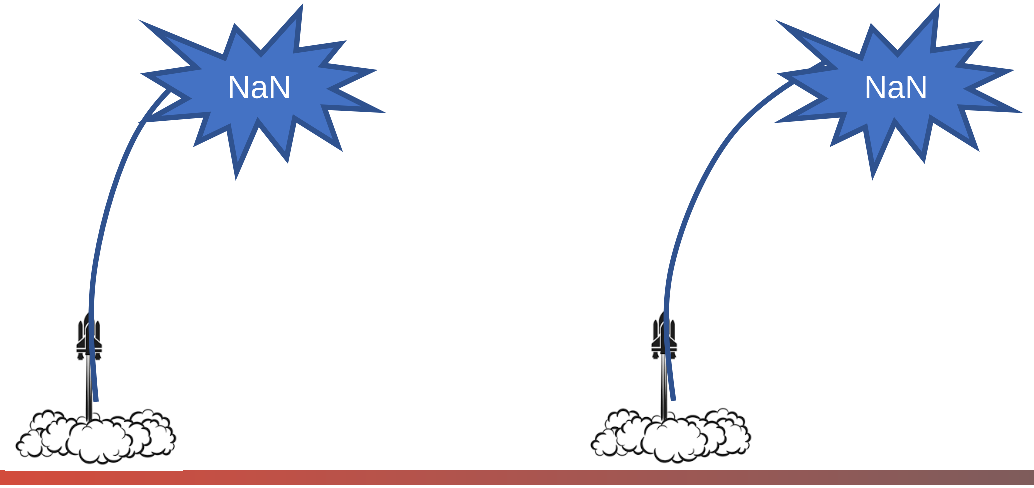
You may need an accurate integrator (complex+costly)

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

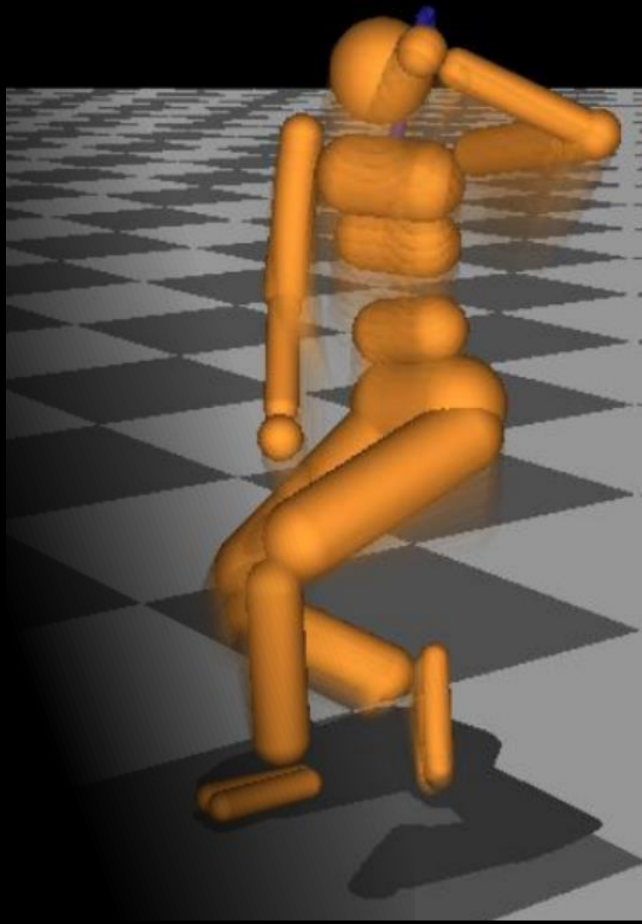
- Easy to implement
  - Integrator, derivatives, Newton-descent
- Side effect: you can focus on efficiency

- 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.

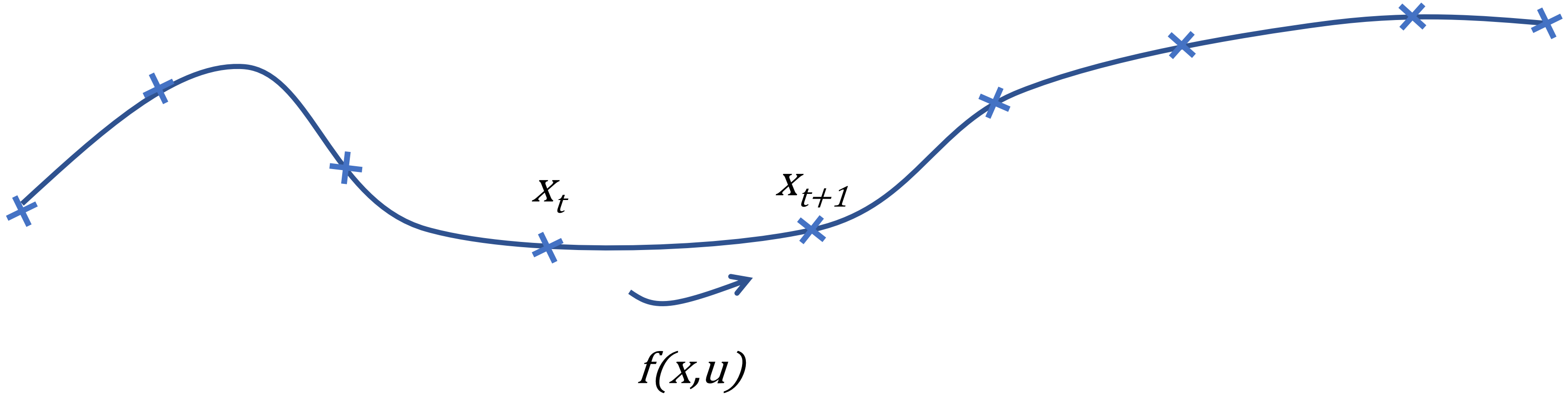


Tassa et al., IROS' 12

Multiple views on DDP

1. DDP as iterative LQR

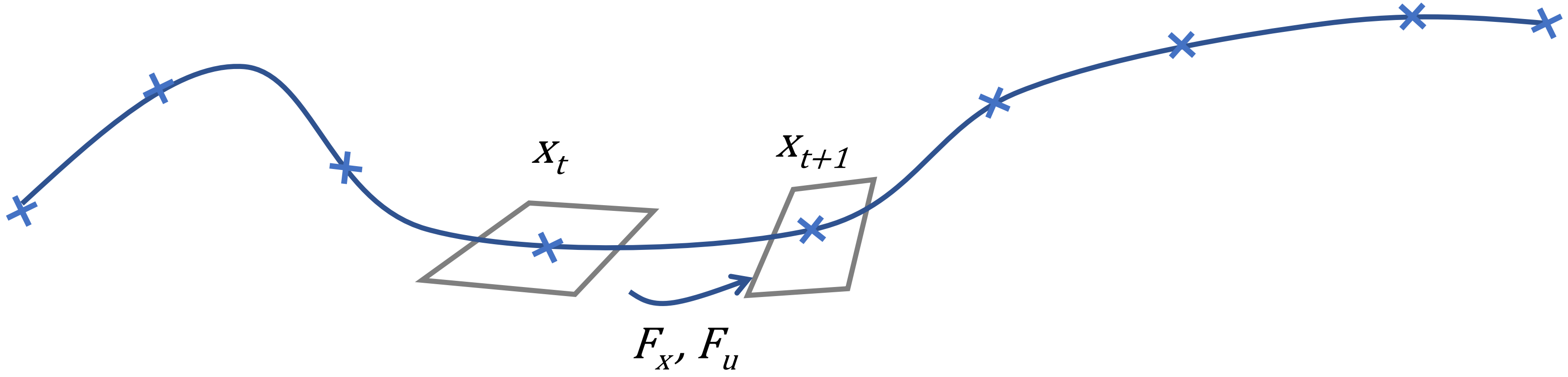
DDP as iterative LQR



- "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 with Ricatti recursion (textbook)

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

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

DDP as iterative LQR

- Algorithm iLQR

```
Initialize with a given trajectory {x_0},{u_0}
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

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

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

DDP as iterative LQR

- Reminder
- Non linear problem

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

- Resulting “linearization”

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

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

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

- Lagrangian on the QP

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

DDP as sparse SQP

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

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

- Lagrangian on the QP

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

- Newton step

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

The sparsity comes from the temporal structure

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

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

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

The sparsity comes from the temporal structure

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

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

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

Stagewise Implementations of Sequential Quadratic Programming for Model-Predictive Control

Armand Jordana*,[1], Sébastien Kleff*,[1], Avadesh Meduri*,[1],
Justin Carpentier[2], Nicolas Mansard[3] and Ludovic Righetti[1]

***Abstract*— The promise of model-predictive control in robotics has led to extensive development of efficient numerical optimal control solvers in line with differential dynamic programming because it exploits the sparsity induced by time. In this work, we argue that this effervescence has hidden the fact that sparsity can be equally exploited by standard nonlinear optimization. In particular, we show how a tailored implementation of sequential quadratic programming achieves state-of-the-art model-predictive control. Then, we clarify the connections between popular algorithms from the robotics community and well-established optimization techniques. Further, the sequential quadratic program formulation naturally encompasses the constrained case, a notoriously difficult problem in the robotics community. Specifically, we show that it only requires a sparsity-exploiting implementation of a state-of-the-art quadratic programming solver. We illustrate the validity of this approach in a comparative study and experiments on a torque-controlled manipulator. To the best of our knowledge, this is the first demonstration of nonlinear model-predictive control with arbitrary constraints on real hardware.**

I. INTRODUCTION

A. Motivation

Model Predictive Control (MPC) has become popular for online robot decision-making. It has shown compelling results with all kinds of robots ranging from industrial manipulators [1], quadrupeds [2]–[4] to humanoids [5], [6]. The general idea of MPC is to formulate the robot motion generation problem as a numerical optimization problem, i.e., a finite horizon Optimal Control Problem (OCP), and solve it online at every control cycle using the current state measurement as the initial state. This receding horizon strategy allows us to adapt the robot behavior as the state of the system and environment change.

In robotics, Differential Dynamic Programming (DDP) [7] is a popular choice to solve OCPs because it exploits the problem's structure well. This advantage has led to a bustling algorithmic development over the past two decades [8]–[20]. In light of the increasing number of variations of DDP, one might naively ask: why not use well-established optimization algorithms [21]? Is there anything special in MPC that cannot

This work was in part supported by the National Science Foundation grants 1932187, 2026479, 2222815 and 2315396.

* Equal contribution - first authors listed in alphabetical order.

[1] Machines in Motion Laboratory, New York University, USA aj2988@nyu.edu, sk8001@nyu.edu, am9789@nyu.edu, lr114@nyu.edu

[2] Inria - Département d'Informatique de l'École normale supérieure, PSL Research University. justin.carpentier@inria.fr

[3] LAAS-CNRS, Université de Toulouse, CNRS, Toulouse. nmansard@laas.fr

be tackled by, for example, an efficient i
Sequential Quadratic Programming (SQP) [
we show that special implementations of
ods developed by the optimization-based c
[23]–[26] are, in fact, sufficient to achie
MPC on real robots.

B. Related work

Mayne first introduced DDP [7] as an
to solve nonlinear OCPs by iteratively
ward pass over the time horizon and a
rollout of the dynamics. This algorithm
linear complexity in the time horizon an
convergence [27]. More recently, Todorov
in DDP by proposing the iterative Linear Q
(iLQR) [8], a variant discarding the seco
the dynamics. It has since gained a lot
the robotics community [3]–[5], [14], [18],
to Gauss-Newton optimization has been
[28]. However, this approach faces two
1) as a single shooting method, it requir
feasible initial guess, which makes the
to warm-start, an essential requirement
tation times [12] and 2) enforcing equali
constraints is not straightforward. The co
to enforce constraints softly using penalty
function. But this approach is heuristic (i.
weight tuning) and tends to cause numeric

Multiple shooting for optimal control, i
addresses the first limitation: it accepts a
guess. Several multiple shooting variants o
proposed in [12], [14] with significantly improved convergence abilities, which have enabled nonlinear MPC at high frequency on real robots [1], [3], [6], [14].

The second issue of enforcing constraints inside a DDP-like algorithm has been addressed in several works. [10] uses a DDP-based projected Newton method to bound control inputs. This approach has further been improved and deployed on a real quadruped robot in [17]. More recently, augmented Lagrangian methods have been used to enforce constraints in iLQR/DDP algorithms [11], [13], [16], [19]. However, their convergence behavior is less understood than DDP, whose seminal paper [7] was followed by sophisticated proofs [27]. To the best of our knowledge, it has not yet been shown that those recent DDP-based algorithms exhibit global convergence (i.e., convergence from any initial point to a stationary point) and quadratic local convergence.

$$
\begin{bmatrix}
\Gamma_1 & M_1^T & 0 & 0 & \cdots & 0 \\
M_1 & \Gamma_2 & M_2^T & 0 & \cdots & 0 \\
0 & M_2 & \Gamma_3 & M_3^T & \cdots & 0 \\
0 & 0 & M_3 & \Gamma_4 & \ddots & 0 \\
\vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\
0 & 0 & 0 & 0 & \ddots & \Gamma_T
\end{bmatrix}
\begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ \vdots \\ s_T \end{bmatrix}
=
\begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \\ \vdots \\ g_T \end{bmatrix}
\tag{7}
$$

where:

$$
\Gamma_k = \begin{bmatrix} R_{k-1} & 0 & -B_{k-1}^T \\ 0 & Q_k & I \\ -B_{k-1} & I & 0 \end{bmatrix}, \quad
M_k = \begin{bmatrix} 0 & S_k^T & 0 \\ 0 & 0 & 0 \\ 0 & -A_k & 0 \end{bmatrix}
$$

$$
\text{and} \quad s_k = \begin{bmatrix} \Delta u_{k-1} \\ \Delta x_k \\ -\lambda_k \end{bmatrix}, \quad
g_k = \begin{bmatrix} -r_{k-1} \\ -q_k \\ \gamma_k \end{bmatrix}. \tag{8}
$$

Sebastien Kleff

Armand Jordana

Avadesh Meduri

Rohan Budhiraja

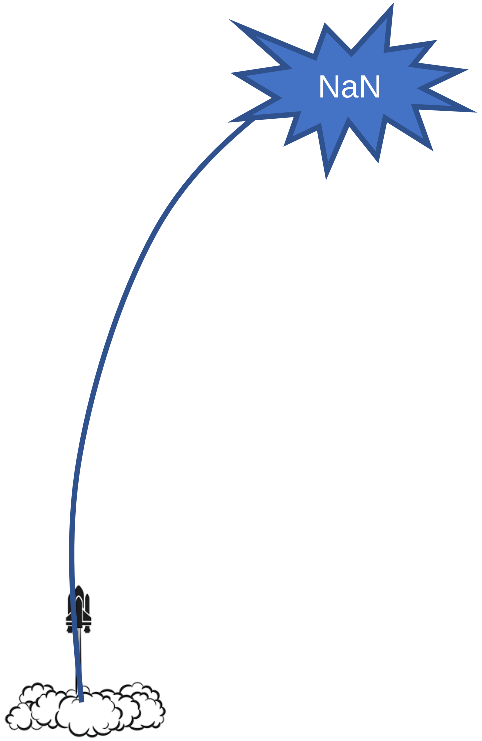
crocoddyl

Contact Robot Optimal Control by Differential Dynamic Library

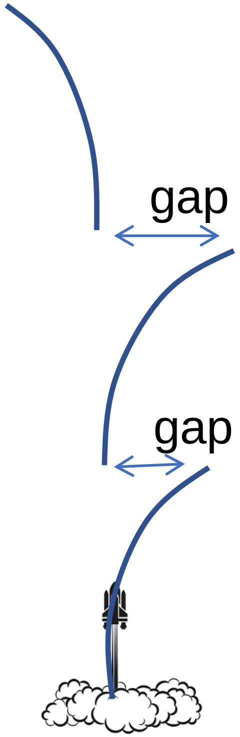
What is Crocoddyl good for, and what is beyond?

BEYOND DDP

Multiple shooting

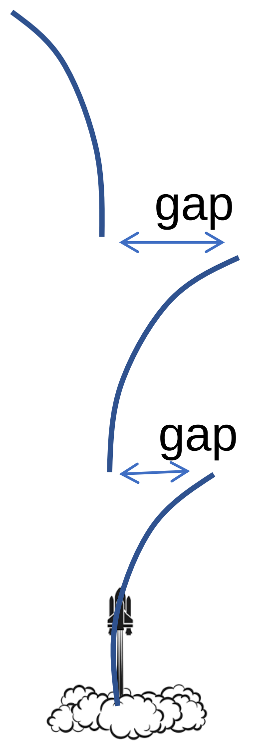


Single shooting
“Your control is bad! “

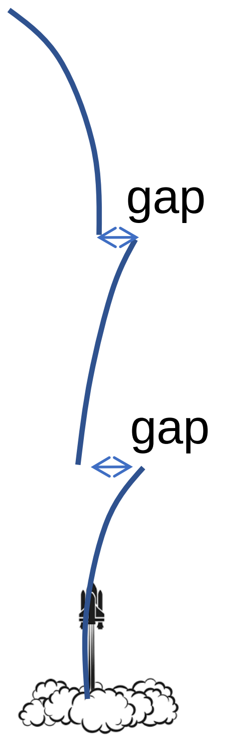


Multiple shooting
“Your control is bad! “

Multiple shooting

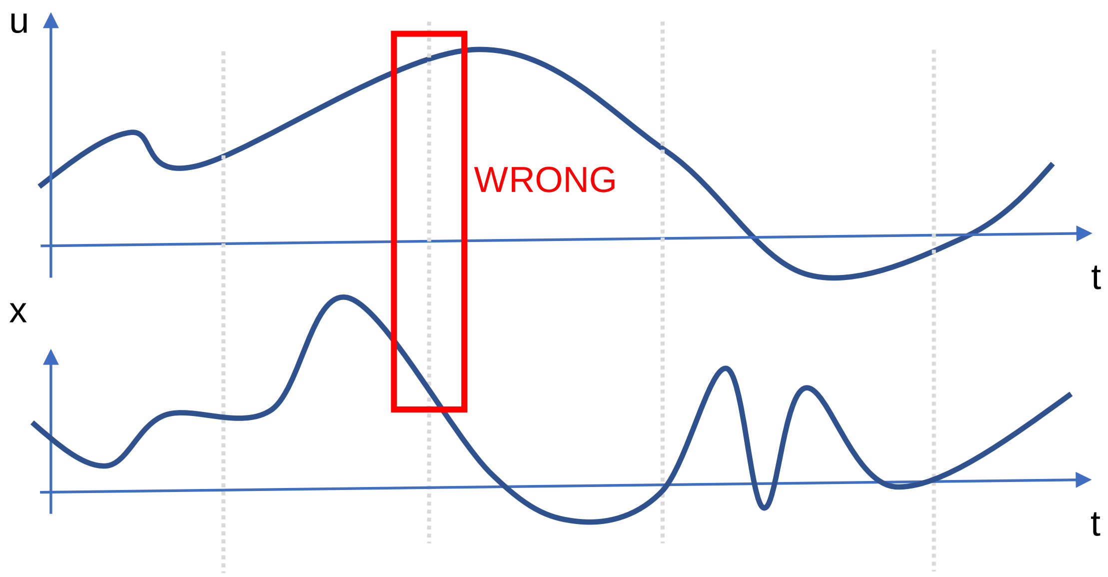


“Your control is bad! “



“Still bad, but better“

Interpretation of dynamics violation



Polynomials(θ_u)

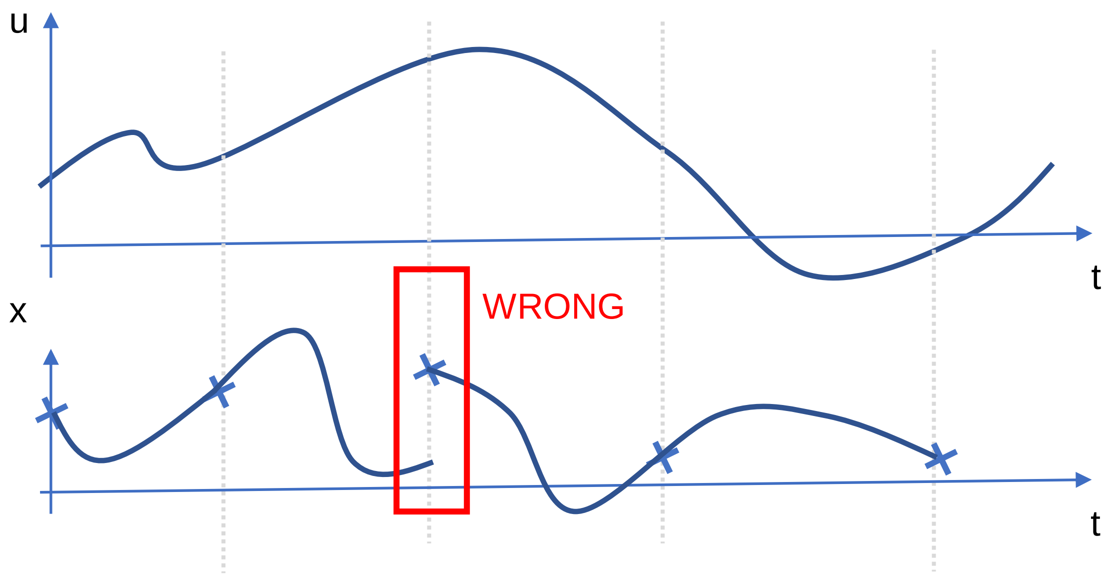
Polynomials(θ_x)

- Collocation:

  We have state and control trajectories

  … and they do not match

Interpretation of dynamics violation



Polynomials(θ_u)

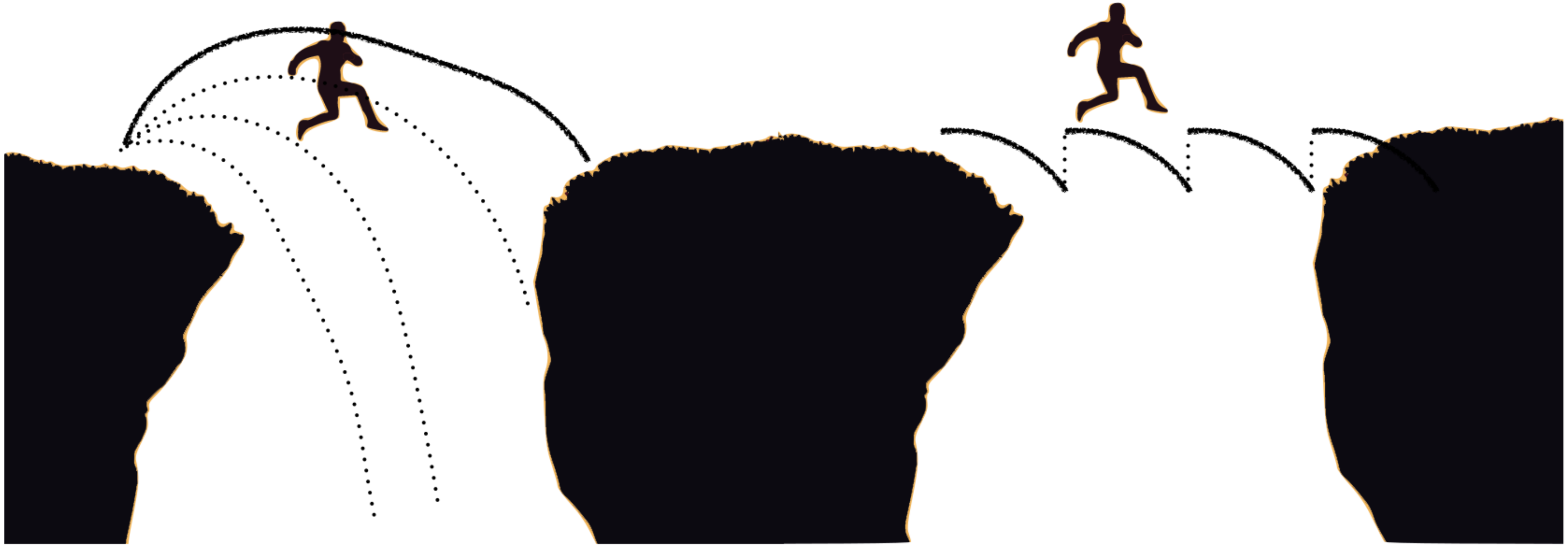
Polynomials(θ_x)

- 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

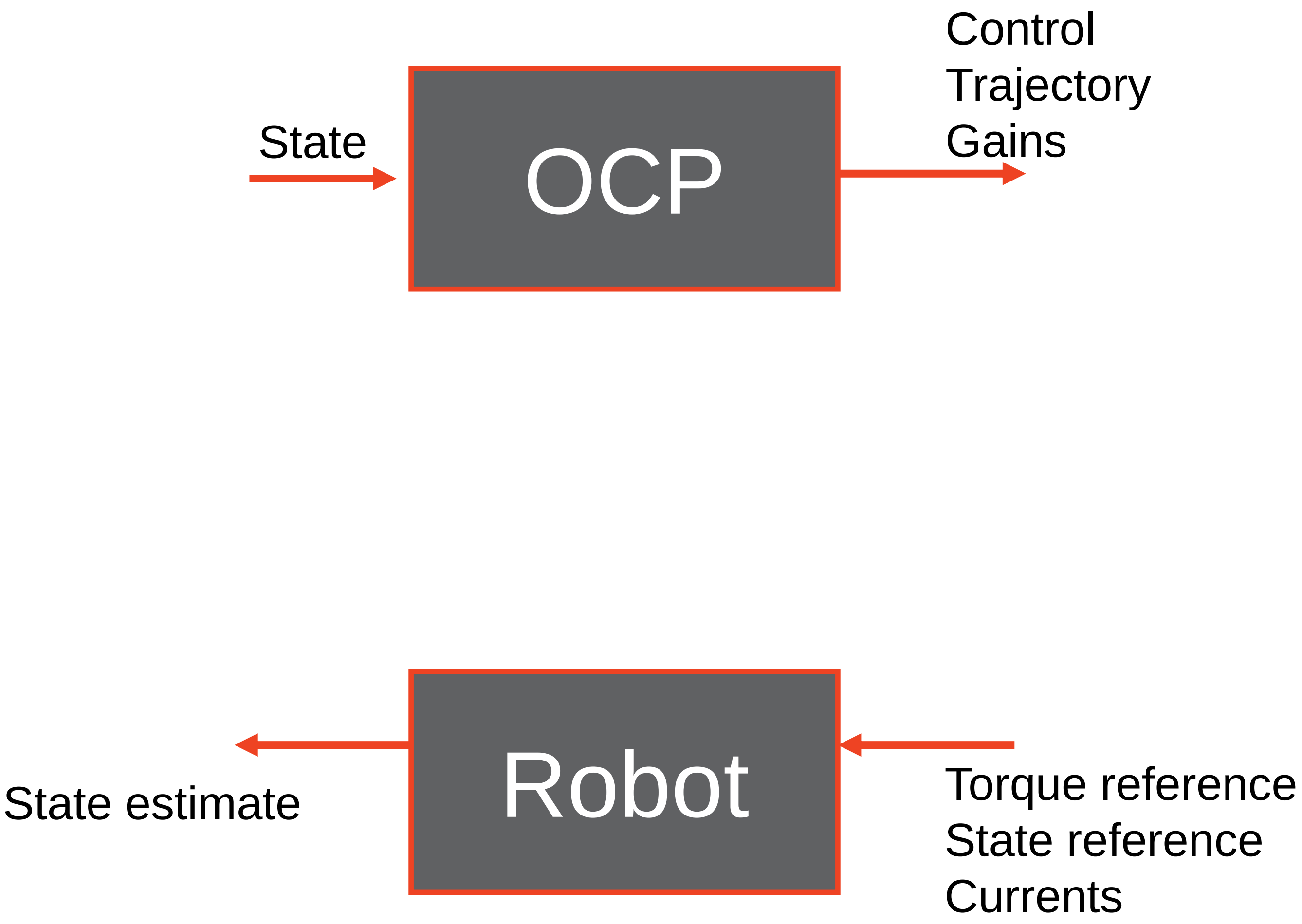
SQP, active set

By penalty

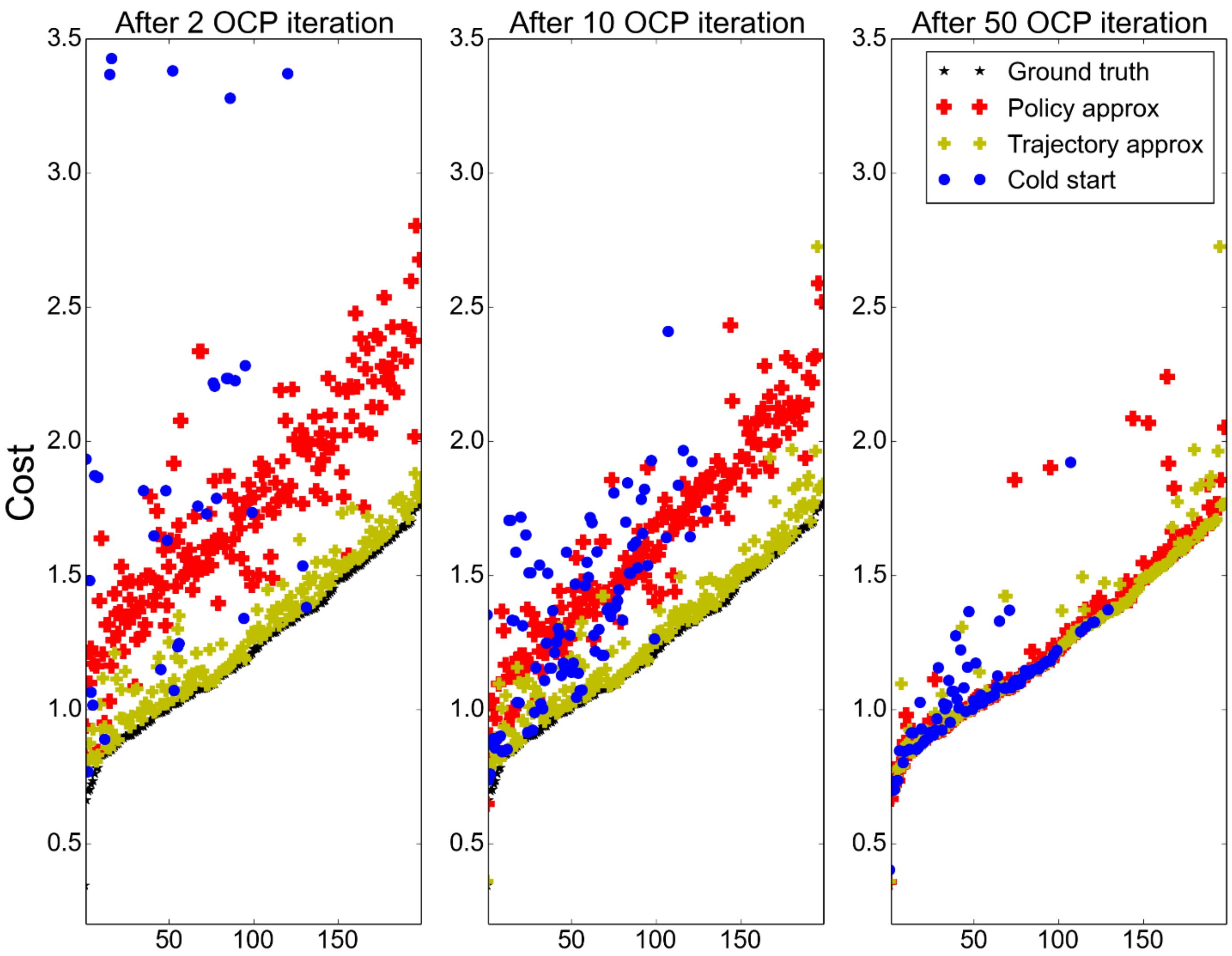
Interior point, augmented lagrangian

Model predictive control

- Closing the loop on the real robot



Importance of the warm start



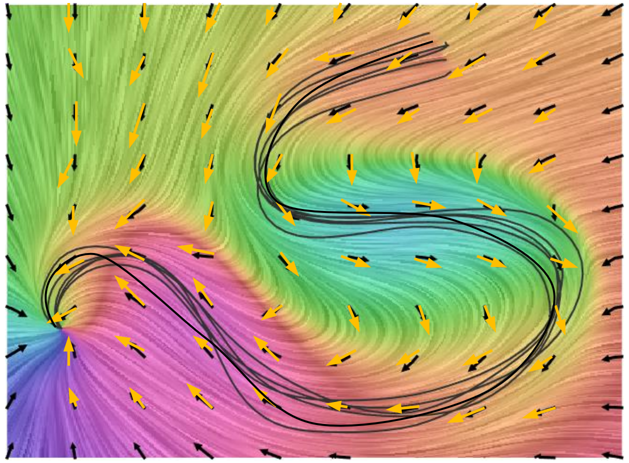
Mansard, N., et al. Using a memory of motion to efficiently warm-start a nonlinear predictive controller. In IEEE ICRA

memmo

$$\min_{\substack{X=(Q,\dot{Q}),\\ U=\tau}} \int_0^T l(x_t,u_t)dt$$

$$\text{s.t.} \quad x(0) = \hat{x},$$

$$\dot{x}(t) = f\big(x(t),u(t)\big), \forall \text{t=0..T}$$



Trajectory optimization

$U: t \longrightarrow u(t)$

Motion planning

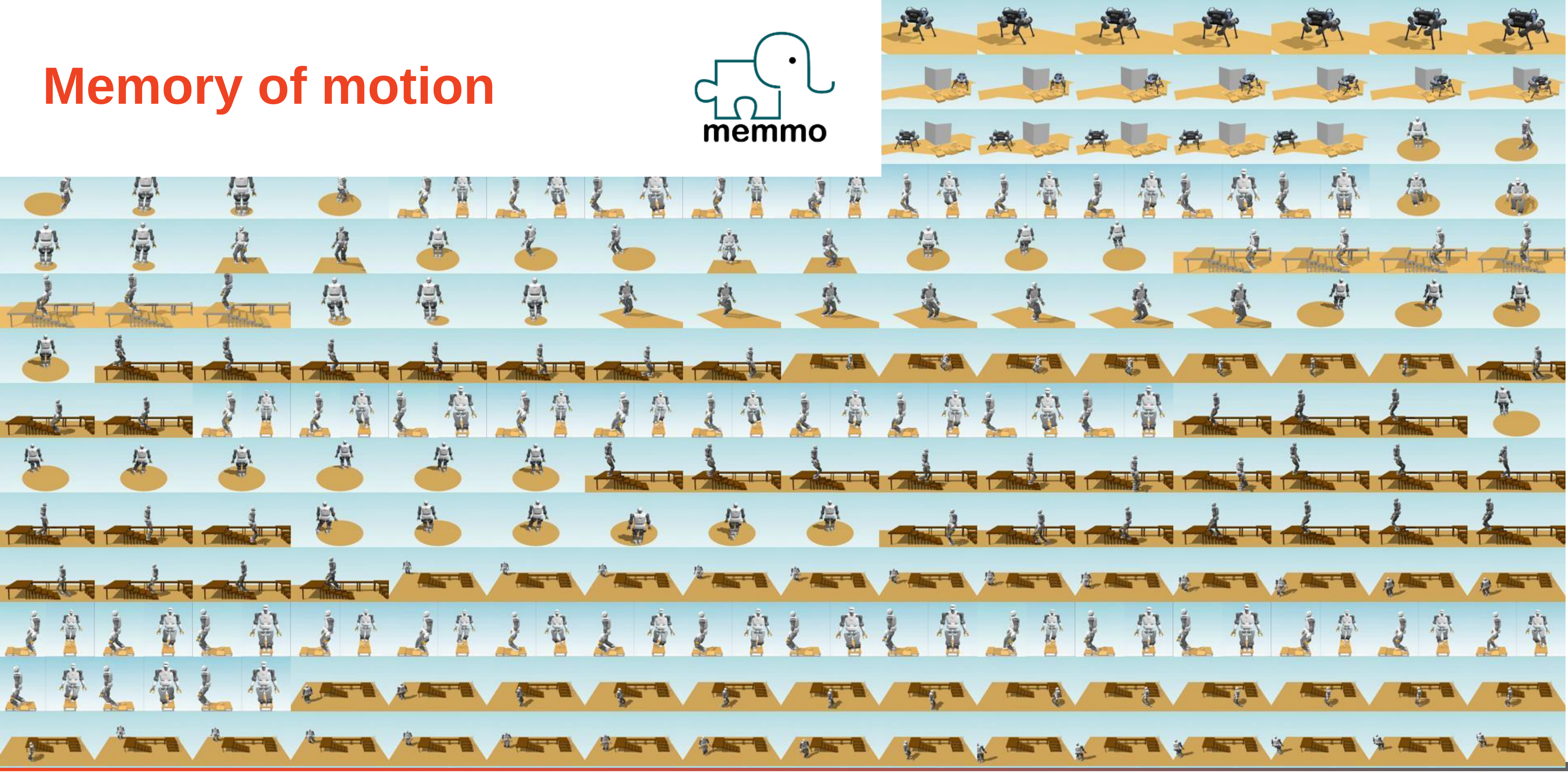
Policy optimization

$\Pi: x \longrightarrow u = \Pi(x)$

Reinforcement learning

Memory of motion

memmo



Time to warm up your fingers!

THE END

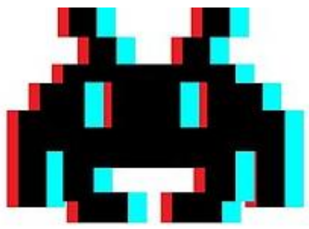
Take-home messages

Numerical problems (few/none discrete constraints)

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

The formulation/transcription is our central problem

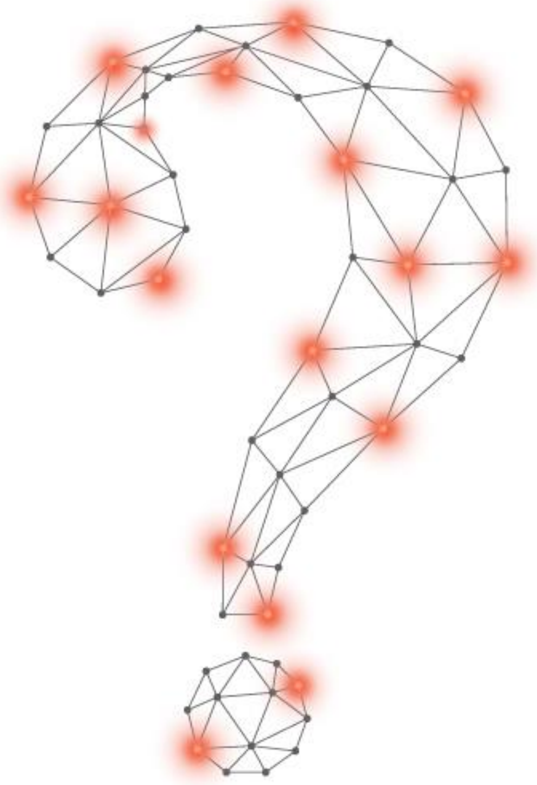
- expert+math knowledge
- keep generalization



Optimal control = reinforcement learning

- train offline
- generalize online

Questions and Answers



Contact Details

Nicolas Mansard

CNRS

nmansard@laas.fr

Funded by the European Union under GA no 101070165.

AGIMUS

Thank you very much for your attention!

www.agimus-project.eu