https://github.com/nmansard/wan24

Optimal control for walking robots

Theory and practice with Crocoddyl

Nicolas Mansard

Gepetto, LAAS-CNRS & ANITI

#03: Transcription, from OCP to NLP

Optimal control problem

$$\underbrace{\min_{\{x\},\{u\}}}_{\text{Find control inputs to minimize cost}} \int_0^T \underbrace{l\big(x(t), u(t)\big)}_{\text{stage costs}} dt + \underbrace{l_T\big(x(T)\big)}_{\text{terminal cost}}$$

$$x_0 = \hat{x}$$ initial dynamics

$$\dot{x}(t) = f(x(t), u(t))$$ deterministic dynamics

Transcribing: *"representing" the reality*

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

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

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

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

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

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

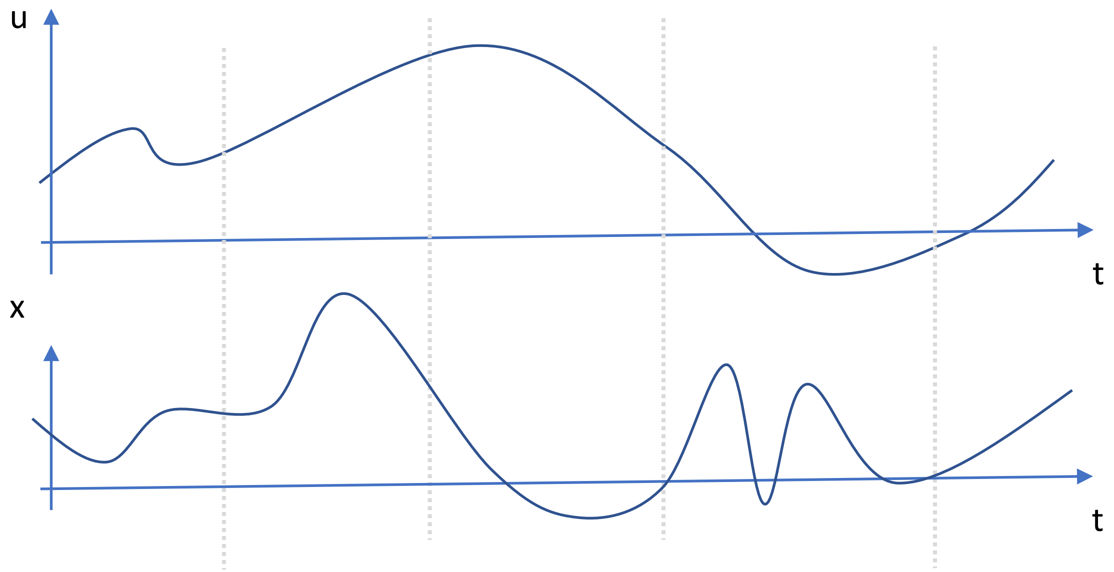
$\theta_x\ \theta_u$ represents the continuous $\underline{x}$,$\underline{u}$ trajectories

Transcribing: *“representing“ the reality*

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

– what about $\{x\}$?

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



Polynomials(θ_u)

Polynomials(θ_x)

Transcribing: *"representing" the reality*

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

– what about $\{x\}$?

- Collocation: $\{x\}$ 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

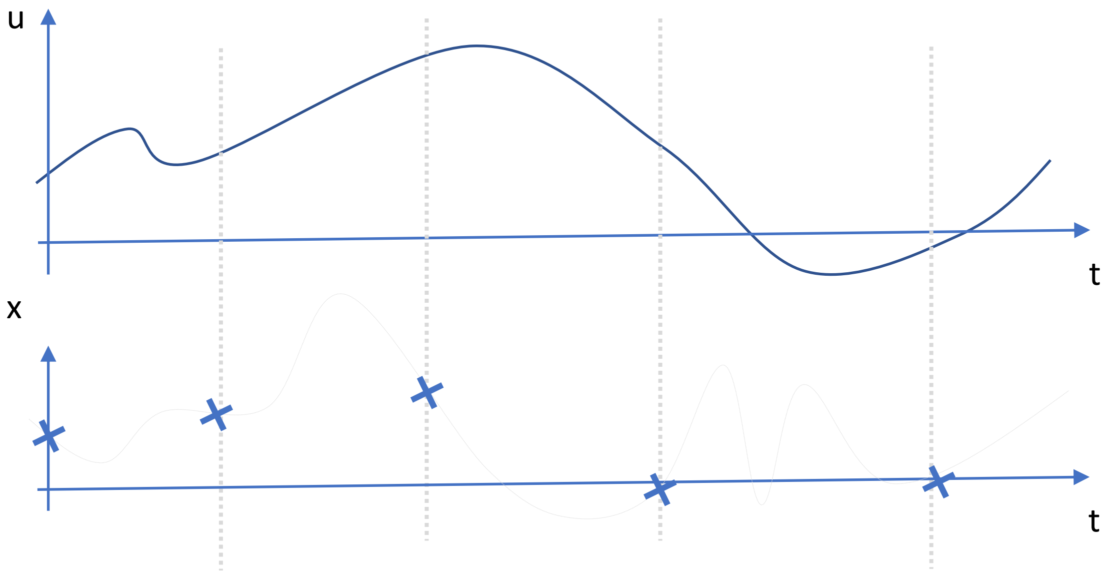
Transcribing: *“representing“ the reality*

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

– what about $\{x\}$?

- Shooting: $\{x\}$ is represented by and integrator

  and only evaluated sparsely



Polynomials(θ_u)

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

Transcribing: *"representing" the reality*

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

– what about $\{x\}$?

- Shooting: $\{x\}$ is represented by and integrator

  and only evaluated sparsely

Problems:

The state is sparsely and approximately known

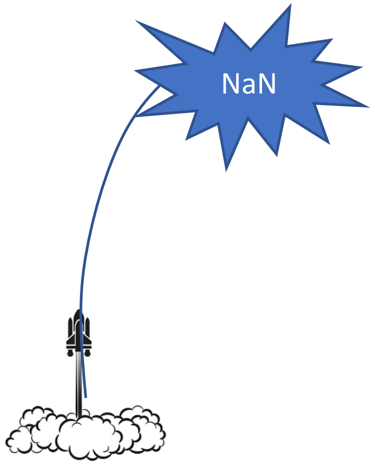
You may need an accurate integrator (complex+costly)

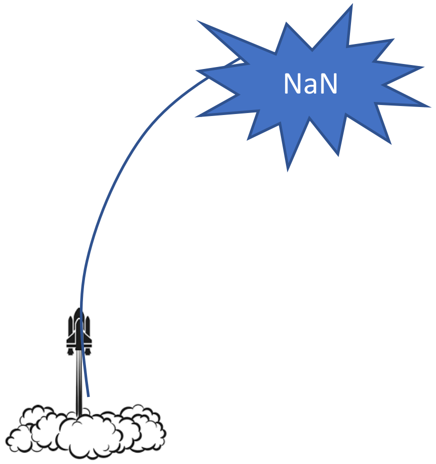
Shooting as control-only problem

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

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

- Unconstrained optimization
- The function $\{u\} \rightarrow \{x\}$ 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 …

Front-end / back-end separation

Discretize first, solve second

- Front end

Formulation of the motion problem, system model, constraints

Discretization

Formulation of a static optimization problem with constraints

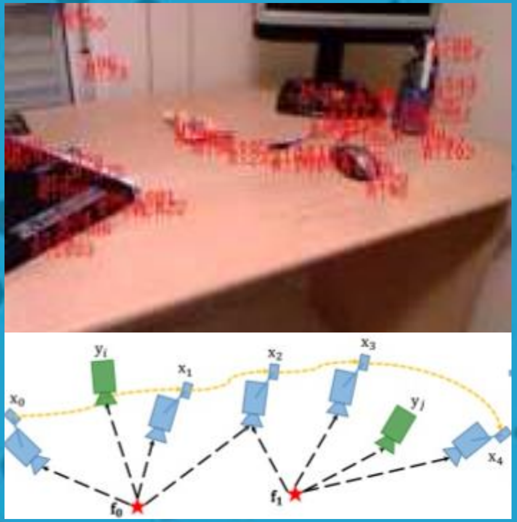
Formulation
+
transcription

- Back end

Resolution of the static optimization problem

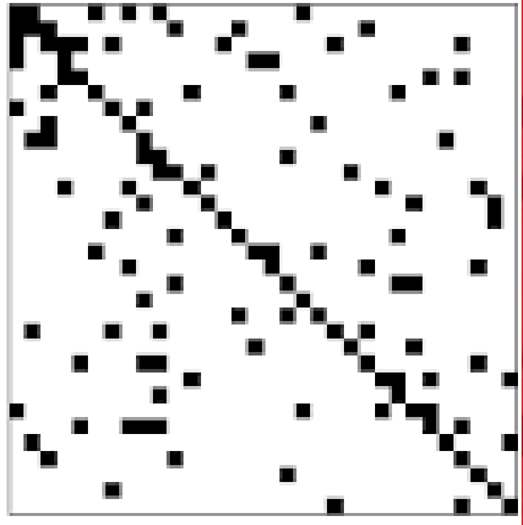
Resolution

Front end



Formulation of a static nonlinear optimization problem (NLP)

Resolution by convex optimization



Back end

Markovian optimal control problems

$$\min_{\{x\},\{u\}} \; l_0(x_0, u_0) + l_1(x_1, u_1) + l_2(x_2, u_2) + \ \ldots + l_T(x_T)$$

s.t.

$$x_0 = x_0{}^{ref}$$

$$\mathrm{f}(x_0, u_0) = x_1$$

$$\mathrm{f}(x_1, u_1) = x_2$$

$$\ldots.$$

$$\mathrm{f}(x_{T-1}, u_{T-1}) = x_T$$

Solving with SQP

$$\min_{y} c(y) \quad s.t. \quad g(y) \geq 0$$

- Lagrangian:

$$\mathcal{L}(y, \lambda) = c(y) - \lambda^T g(y)$$

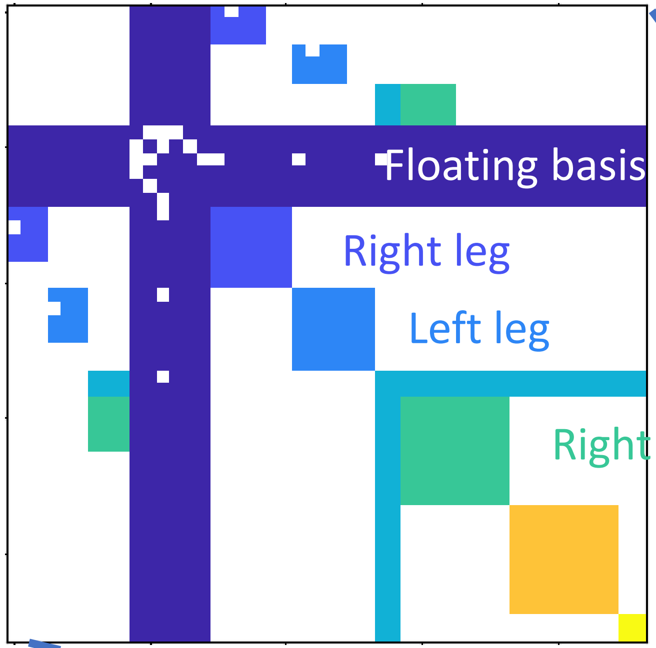
- Solving with Newton method:

$$\nabla^2 \mathcal{L} = \begin{pmatrix} \nabla^2 c - \lambda^T \nabla^2 g & \nabla g^T \\ \nabla g & 0 \end{pmatrix}$$

Resulting KKT system

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

… for efficient problems



Justin Carpentier
Inria Paris / PR[AI]RIE

$$
\left[\begin{array}{cccc|ccc|cccc}
L_{xx} & & & & L_{xu} & & & -I & F_x^T & & \\
 & \ddots & & & & \ddots & & & \ddots & \ddots & \\
 & & L_{xx} & & & & L_{xu} & & & -I & F_x^T \\
 & & & L_{xx} & & & & & & & -I \\
\hline
L_{ux} & & & & L_{uu} & & & & F_u^T & & \\
 & \ddots & & & & \ddots & & & & \ddots & \\
 & & L_{ux} & & & & L_{uu} & & & & F_u^T \\
\hline
-I & & & & & & & & & & \\
\boxed{F_x} & -I & & & F_u & & & & & & \\
 & \ddots & \ddots & & & \ddots & & & & & \\
 & & F_x & -I & & & F_u & & & &
\end{array}\right]
$$

Pinocchio
https://github.com/stack-of-tasks/pinocchio

BSD
BSD-2 License

Carpentier et al. (2019) – https://hal.science/hal-01866228v2