Numerical Optimal Control

Enrico Bertolazzi

DII - Department of Industrial Engineering – University of Trento

June 5, 2015

Outline

OCP definition
Simplified definition

An optimal control problem is a constrained optimisation problem with a dynamical system as constraint.

- Dynamical system:

$$\boldsymbol{x}'(t) = \boldsymbol{f}\left(\boldsymbol{x}(t), \boldsymbol{u}(t), t\right), \qquad t \in (0, T)$$

  with initial condition $\boldsymbol{x}(0) = \boldsymbol{x}_0$.
  - $\boldsymbol{x}(t) \in \mathbb{R}^n$: the *states* of the dynamical system
  - $\boldsymbol{u}(t) \in \mathcal{U} \subset \mathbb{R}^m$: the *controls* (piecewise continuous)
  - If $\boldsymbol{u}(t)$ and $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t)$ are regular enough has an unique solution.
- The *target* or the *performance functional index*:

$$J(\boldsymbol{u}) = \underbrace{\mathcal{M}(\boldsymbol{x}(T))}_{\text{Mayer term}} + \int_0^T \underbrace{\ell(\boldsymbol{x}, \boldsymbol{u}, t)}_{\text{Lagrange term}} \, \mathrm{d}t$$

- The OCP: find $\boldsymbol{u}$ that minimizes the *target* subject to the *dynamical system*

OCP definition
Complications

- States and controls may be constrained:

$$\boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \geq \mathbf{0}, \quad \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) = \mathbf{0}$$
$$\int_0^T \boldsymbol{e}(\boldsymbol{x}(t), \boldsymbol{u}(t), t)\mathrm{d}t = \mathbf{0},$$

- T may be fixed or part of the problem of minimization.
- Constants parameters may be part of the problem, e.g.

$$\boldsymbol{x}'(t) = \boldsymbol{f}\left(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, t\right),$$

  where $\boldsymbol{p} = (p_1, p_2, \ldots, p_r)$.
- Problem may contains interfaces, e.g.

$$\boldsymbol{x}'(t) = \boldsymbol{f}_k\left(\boldsymbol{x}(t), \boldsymbol{u}(t), t\right), \quad t \in (t_{k-1}, t_k) \qquad \boldsymbol{x}^+(t_k) = \Phi(\boldsymbol{x}^-(t_k), t_k),$$

OCP definition
Simpification

Some complication may be reduced by simple tricks

- Integral constraints may be eliminated by adding states

$$\int_0^T e_k(\boldsymbol{x}(t), \boldsymbol{u}(t))\mathrm{d}t = 0 \quad \Rightarrow$$

$$z'(t) = e_k(\boldsymbol{x}(t), \boldsymbol{u}(t)), \qquad z(0) = z(T) = 0$$

- Constants parameters may be treated as additional constants stated, e.g. p_k is a parameter define $p_k(t)$ as a new state:

$$p_k'(t) = 0,$$

- If T is free by a change of variable $t = \xi T$ the problem is transformed to a fixed boundary problem in the interval $[0, 1]$.

OCP definition
Simpification (slack variables)

- Inequalities

$$g_k(\boldsymbol{x}(t), \boldsymbol{u}(t)) \geq 0 \qquad \Rightarrow \qquad g_k(\boldsymbol{x}(t), \boldsymbol{u}(t)) = \varepsilon_k(t), \quad \varepsilon_k(t) \geq 0$$

- Controls are better treated if are bounded in a ipercube

$$u_{k,\min} \leq u_k(t) \leq u_{k,\max}$$

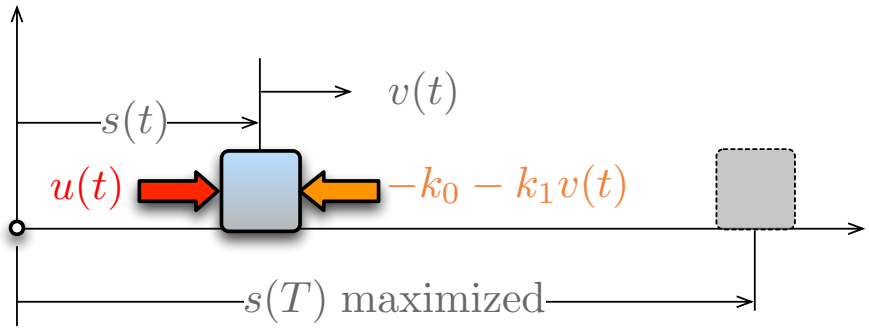
- Dynamical system may have more complex BC than $\boldsymbol{x}(0) = \boldsymbol{x}_0$, e.g. $\boldsymbol{b}(\boldsymbol{x}(0), \boldsymbol{x}(T)) = \mathbf{0}$:

$$\begin{cases} \boldsymbol{b}(\boldsymbol{x}(0), \boldsymbol{x}(T)) = \mathbf{0}, \\ \partial^T_{\boldsymbol{x}(0)} \boldsymbol{b}(\boldsymbol{x}(0), \boldsymbol{x}(T)) \cdot \boldsymbol{\omega} = \boldsymbol{\lambda}(0), \\ \partial^T_{\boldsymbol{x}(T)} \boldsymbol{b}(\boldsymbol{x}(0), \boldsymbol{x}(T)) \cdot \boldsymbol{\omega} = -\partial^T_{\boldsymbol{x}} \mathcal{M}(\boldsymbol{x}(T)) - \boldsymbol{\lambda}(T), \end{cases}$$

  additional Lagrange multiplier are introduced.

Demonstration example

Point mass maximum travel



Find the force u that moves a mass to the longest distance (T fixed):

$$\min_{u \in \mathcal{U}} \int_0^T (-v(t)) \mathrm{d}t = \min_{u \in \mathcal{U}} \left(s(0) - s(T) \right)$$

subject to:

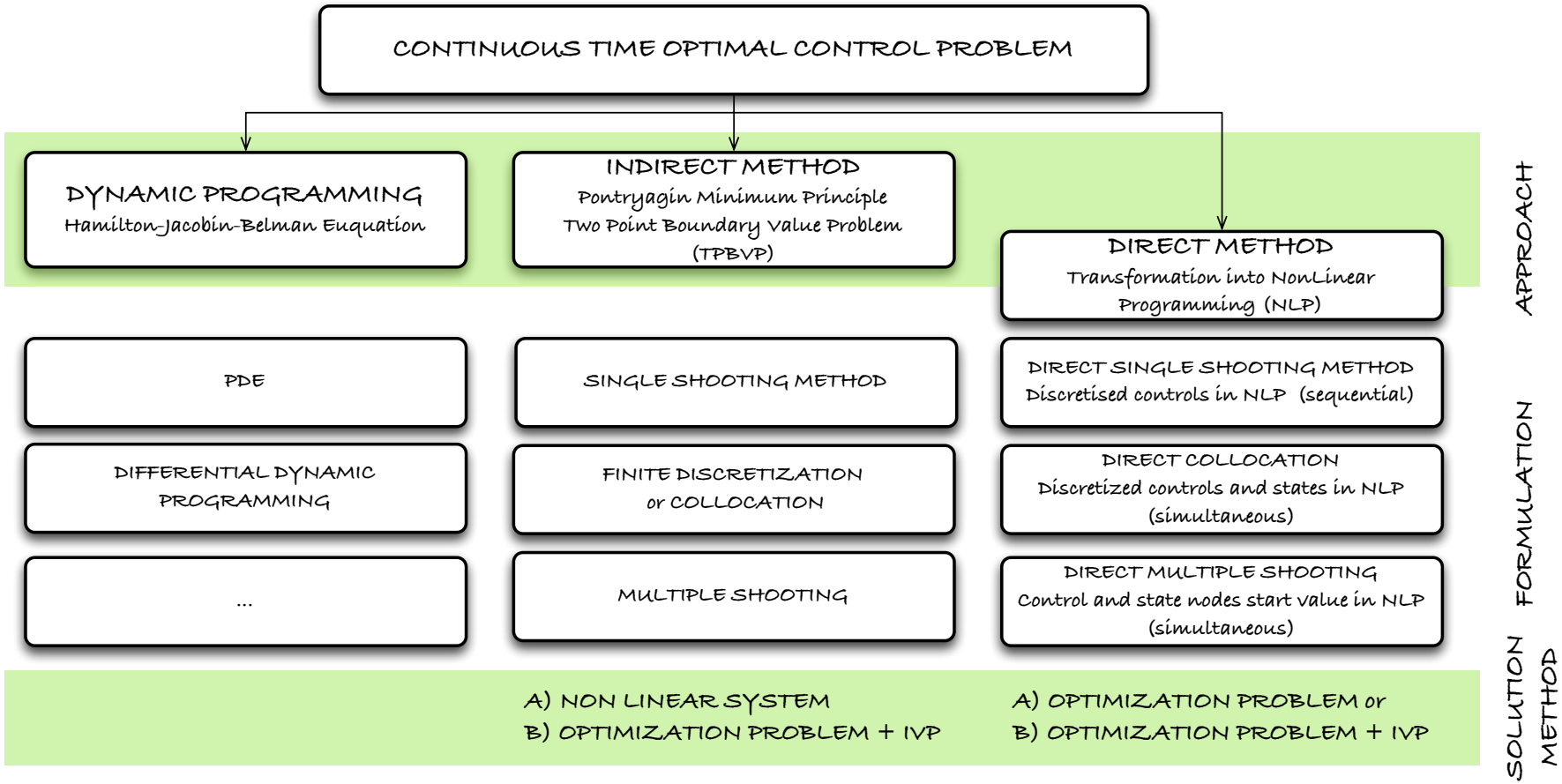
$$s'(t) = v(t), \quad v'(t) = u(t) - k_0 - k_1 v(t) - k_2 v(t)^2 \qquad \text{system dynamics}$$

$$s(0) = 0, \quad v(0) = 0, \quad v(T) = 0 \qquad \text{boundary conditions}$$

$$|u(t)| \leq g + k_3 v(t)^2 \qquad \text{control bounds}$$

General overview of Solution Methods

Classification of solution methods



Dynamic Programming

Principle of Optimality

Each sub-trajectory of an optimal trajectory is an optimal trajectory.

This provides a solution method for Hamilton-Jacobi-Bellman (HJB) equation:

$$-\frac{\partial}{\partial t}J(\boldsymbol{x},t)=\min_{u\in\mathcal{U}}\left\{\ell(\boldsymbol{x},\boldsymbol{u},t)+\nabla_{\boldsymbol{x}}J(\boldsymbol{x},t)\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u},t)\right\}$$

that can be solved on a shortened horizon starting from the end $J(\boldsymbol{x},T)=M(\boldsymbol{x}_f)$ and recursively find the complete solution backward.

$J(\boldsymbol{x},t)$: *value function* is the cost-to-go to the end when starting at a given state.

The optimal feedback control $\boldsymbol{u}_{fb}$ for the state $\boldsymbol{x}$ at time t is then obtained from:

$$\boldsymbol{u}_{fb}(\boldsymbol{x},t)=\arg\min_{\boldsymbol{u}\in\mathcal{U}}\left\{\ell(\boldsymbol{x},\boldsymbol{u},t)+\nabla_{\boldsymbol{x}}J(\boldsymbol{x},t)\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u},t)\right\}.$$

Pontryagin Minimum Principle

Necessary conditions for optimality

Let us define $\mathcal{H} = \ell(\boldsymbol{x}, \boldsymbol{u}, t) + \boldsymbol{\lambda} \cdot \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t)$ the first variation provides the following necessary conditions (**B**oundary **V**alue **P**roblem, BVP):

$$\begin{aligned} \boldsymbol{x}'(t) &= \partial_{\lambda}^{T} \mathcal{H}\left(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, t\right) \\ \boldsymbol{\lambda}'(t) &= -\partial_{\boldsymbol{x}}^{T} \mathcal{H}\left(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, t\right) \\ \boldsymbol{x}(0) &= \boldsymbol{x}_0, \qquad -\boldsymbol{\lambda}(T) = \partial_{\boldsymbol{x}_f}^{T} \mathcal{M}(\boldsymbol{x}_f) \end{aligned}$$

and a local optimisation of the Hamiltonian at each time instant:

$$\boldsymbol{u}_{fb}(\boldsymbol{x}, \boldsymbol{\lambda}, t) = \arg\min_{\boldsymbol{u} \in \mathcal{U}} \left\{ \ell(\boldsymbol{x}, \boldsymbol{u}, t) + \boldsymbol{\lambda} \cdot \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) \right\}.$$

or when $\mathcal{U} = \mathbb{R}^m$

$$\boldsymbol{u}(\boldsymbol{x}, \boldsymbol{\lambda}, t) \text{ is a solution of } \partial_{\boldsymbol{u}}^{T} \mathcal{H}\left(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, t\right) = \boldsymbol{0},$$

Pontryagin Minimum Principle
Solution Methods

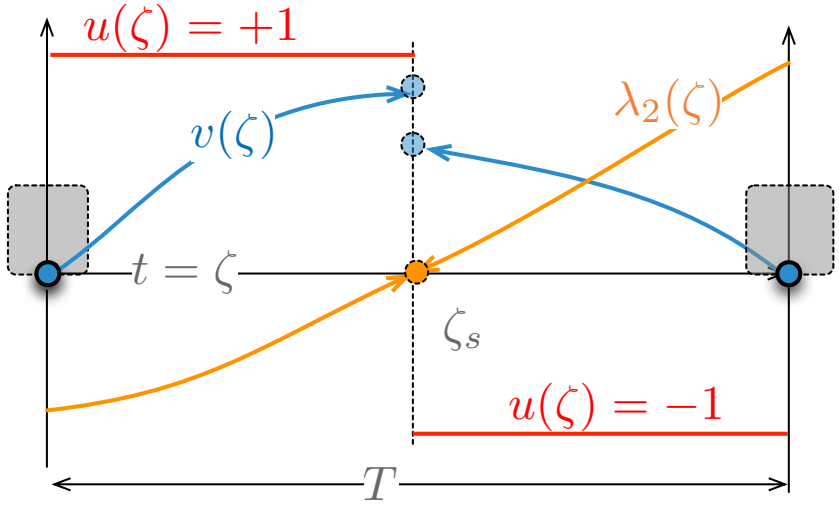
- Analytical solution: possible for simple cases
- Numerical solution:
  - finite difference approximation of BVP + minimisation (optional)
  - approximation of BVP via collocation + minimisation (optional)
  - single shooting + minimisation
  - multiple shooting + minimisation
  - ...

Demo Example with PMP
Analytical solution 1/3

From first variation we get the following BVP:

Dynamical system equations:

$$x' = v$$
$$v' = u - k_0 - k_1 v - k_2 v^2$$



Adjoint equations:

$$\lambda_1' = 0$$
$$\lambda_2' = 1 - \lambda_1 + \lambda_2 \left(k_1 + 2k_2 v\right)$$

Boundary conditions:

$$x(0) = v(0) = v(T) = \lambda_1(T) = 0$$

Optimal control law:

$$u(v, \lambda_2) = -\left(g + k_3 v^2\right) \operatorname{sign}(\lambda_2)$$

Demo Example with PMP

Analytical solution 2/3

Integrate forward from left and backward from right and match interface conditions, we get switching point (for $k_2 = k_3 = 0$):

$$t_s = \frac{1}{k_1} \ln \frac{g - k_0 + (g + k_0)e^{-k_1 T}}{2g}$$

Then we get the analytical solution:

$$s(t) = \frac{1}{k_1^2} \begin{cases} \left(k_1 t + \mathrm{e}^{-k_1 t} - 1\right)(g - k_0) & t \le t_s \\ g\left(1 + k_1(2T - t) + 2\ln \frac{e^{-k_1 T}(g - k_0) + g + k_0}{2g}\right) & \text{otherwise} \end{cases}$$

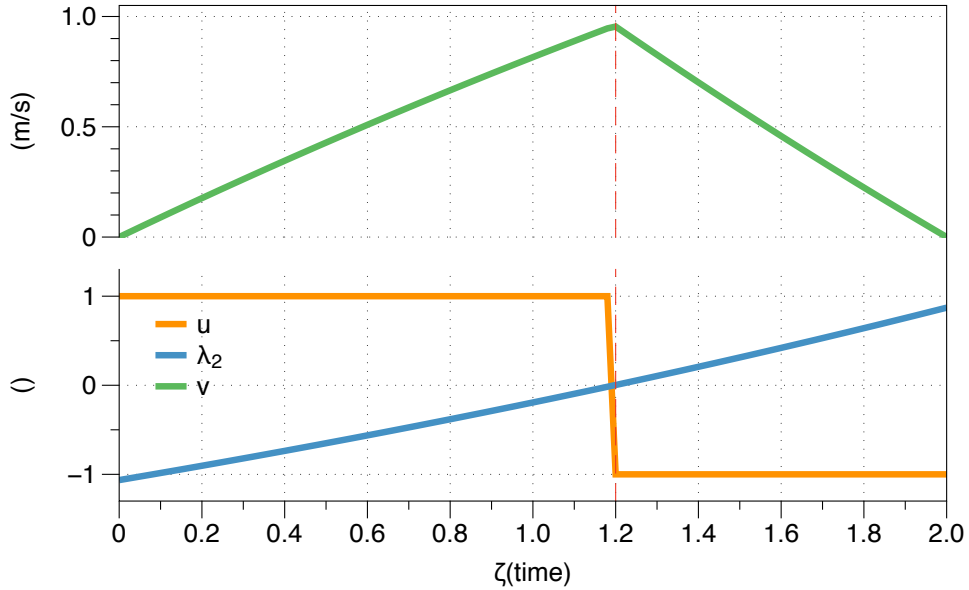
$$v(t) = \frac{1}{k_1} \begin{cases} \left(\mathrm{e}^{-k_1 t} - g\right)(k_0 - g) & t \le t_s \\ (e^{k_1(T-t)} - 1)(k_0 + g) & \text{otherwise} \end{cases}$$

$$\lambda_1(t) = 0, \quad \lambda_2(t) = \frac{1}{k_1}\left(\frac{2\,g\,\mathrm{e}^{k_1(t-T)}}{(g - k_0)\mathrm{e}^{-k_1 T} + k_0 + g} - 1\right)$$

Demo Example with PMP

Analytical solution 3/3

The exact solution for an horizon of $T = 2s$, $k_0 = 0.1, k_1 = 0.2, k_2 = 0, k_3 = 0$:



The optimal control is:

$$u(t) = \begin{cases} 1 & t \leq t_s \\ -1 & t_s < t \leq T \end{cases}$$

Analytical solution with PMP
A general approach

Analytical solutions can be found even for system with more switching points (when their number is known). However process can be complex.

Partial knowledge of exact solution may be used in the discretization process.

Non Linear Programming (NLP)
Direct (Transcription) Methods

The original optimal control problem is discretized and *transcribed* to a Non Linear Programming (NLP).
The NLP is solved using well-established optimization methods.
Methods differs for the variables to be discretized (i.e. control and states) and how to approximate the continuous time dynamics.

- **single shooting**, **multiple shooting**
  only controls are parameterized. ODE solvers + sensitivity analysis required
- **collocation**
  states and controls are parameterized with polynomial functions.
  If orthogonal polyniomial are used: ***pseudo-spectral*** methods.

Available software based on indirect methods are ACADO, GPOPS-III, PSOPT, MISER, SOCS, DIRCOL, PROPT, RIOTS.

Demo example with NLP

Direct transcription with finite difference

Original problem is transcribed into a NLP:

$$\min_{u_k, k=0,\dots,N-1} \sum_{k=0}^{N-1} -\frac{v_{k+1}+v_k}{h}$$

subject to:

$$\frac{s_{k+1}-s_k}{h} = \frac{v_{k+1}+v_k}{2} = \bar{v}_{k+1/2}$$

$$\frac{v_{k+1}-v_k}{h} = -k_0 - k_1\bar{v}_{k+1/2} - k_2\bar{v}^2_{k+1/2} + u_{k+1/2}$$

$$s_0 = 0, \qquad v_0 = 0, \qquad v_N = 0$$

$$-g - k_3 v^2_{k+1/2} \le u_{k+1/2} \le 1 + k_3 v^2_{k+1/2}, \quad k = 0, \dots, N-1$$

where $h = \frac{T}{N}$. SQP (Sequential Quadratic Programming) method is used to solve the problem.

Demo example with NLP

Direct transcription with finite difference

This problem can be solved using available NLP solver. For reference the state of art nonlinear optimization code are

- IPOPT
- KNITRO
- LOQO
- WORHP

In this case IPOPT was used to find the numerical solution via its Matlab interface.

Demo example with NLP

Direct transcription with finite difference

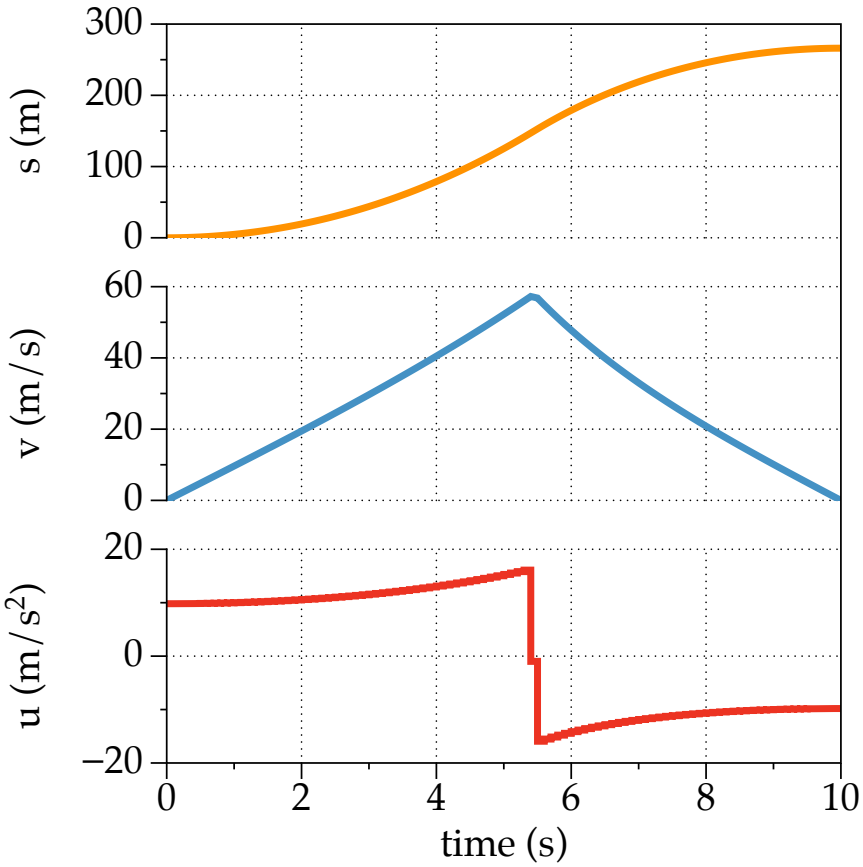


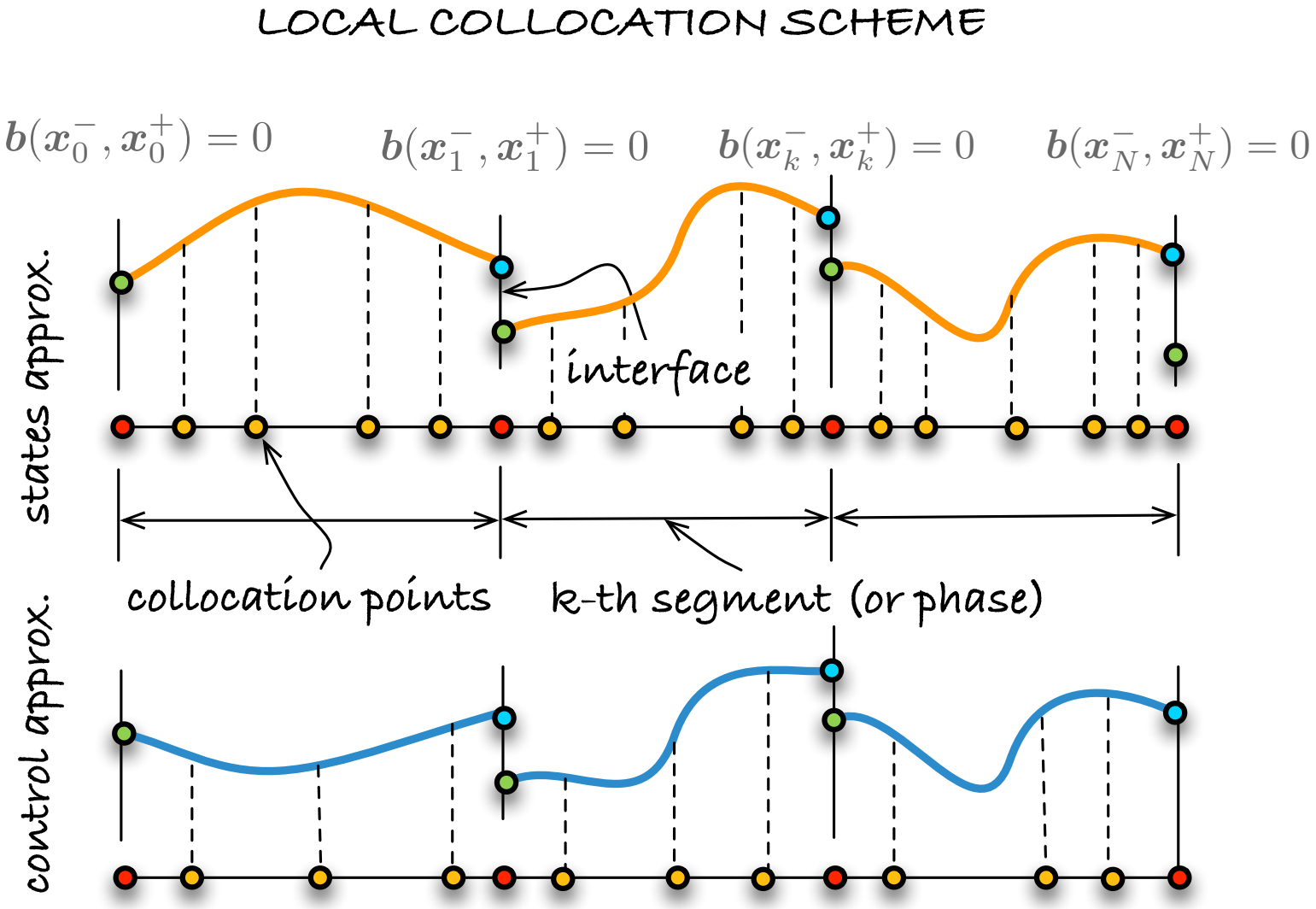
Figure: NLP solution using IPOPT in Matlab with $N = 100$ and also providing the jacobian. Dent in the control solution at the jump location is due to control discretisation.

Demo example with NLP

NLP vs PMP with Finite Difference Approximation

- NLP solves a smaller problem (if NLP method does not use lagrange multipliers)
- NLP naturally treat inequalities
- NLP can use robust well developed optimization software
- NLP is fast (0.02s for Demo problem solution)
- NLP is less accurate

Demo example with NLP
Local collocation



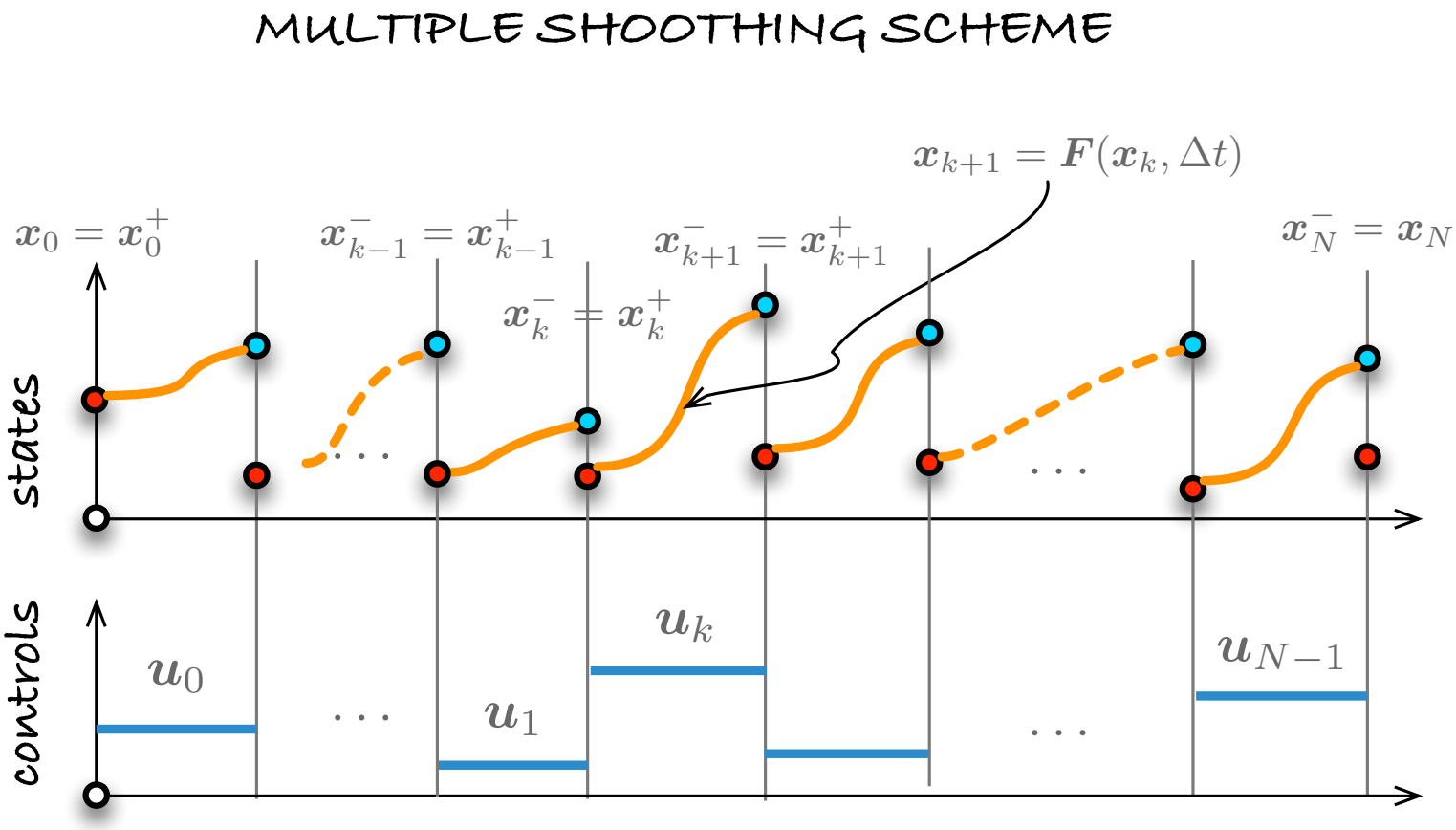
Direct Method: collocation

Brief description

A specific variant of implicit RK methods is called collocation and is derived as follows:

- solution $\boldsymbol{x}$ is approximated by polynomial of $m - th$ order in the collocation interval $t \in [t_k, t_{k+1}]$.
- a grid of collocation points $t(i) = t_k + c_i \Delta t$ is chosen as above, using $0 \leq c_1 < c_2 \cdots < c_m \leq 1$.
- The $(m+1)$ unknown vector coefficients $v_0, \ldots, v_m$ are determined via the $(m+1)$ vector equations that require that the polynomial starts at $\boldsymbol{x}_k$ and its derivative at each collocation point matches the function f at the corresponding value of the polynomial.

Demo example with NLP
Multiple shooting



Demo example with NLP
Multiple shooting: discussion

It is called **Sequential Approach** because the simulation problem and optimization problem are solved sequentially, one after the other.

- all states are eliminated via forward integration (*i.e.* Runge Kutta)
  Minimum number of variables: controls $\boldsymbol{u}_k$ and interface states $\boldsymbol{x}_k$
- sensitivity analysis is necessary to compute derivatives of interface conditionsw $\boldsymbol{x}_k^- = \boldsymbol{x}_k^+$ w.r.t. interface state.
- Interface conditions may be more general $\boldsymbol{x}_k^+ = \boldsymbol{\Phi}(\boldsymbol{x}_k^-)$

Demo example with PMP
Solution with Finite Difference 1/3

Finite difference discretization of the BVP from Pontryagin max principle:

$$\frac{\mathrm{d}}{\mathrm{d}t}s(t_{k+1/2}) \approx \frac{s_{k+1} - s_k}{h} \qquad s(t_{k+1/2}) \approx \bar{s}_{k+1/2} = \frac{s_{k+1} + s_k}{2}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}v(t_{k+1/2}) \approx \frac{v_{k+1} - s_k}{h} \qquad v(t_{k+1/2}) \approx \bar{v}_{k+1/2} = \frac{v_{k+1} + v_k}{2}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\lambda_1(t_{k+1/2}) \approx \frac{\lambda_{1_{k+1}} - \lambda_{1_k}}{h} \qquad \lambda_1(t_{k+1/2}) \approx \bar{\lambda}_{1,k+1/2} = \frac{\lambda_{1_{k+1}} + \lambda_{1_k}}{2}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\lambda_2(t_{k+1/2}) \approx \frac{\lambda_{2_{k+1}} - \lambda_{2_k}}{h} \qquad \lambda_2(t_{k+1/2}) \approx \bar{\lambda}_{2,k+1/2} = \frac{\lambda_{2_{k+1}} + \lambda_{2_k}}{2}$$

Demo example with PMP
Solution with Finite Difference 2/3

Discretized BVP with *explicit* control law:

$$\begin{cases} s_{k+1} = s_k + h\bar{v}_{k+1/2} \\ v_{k+1} = v_k + h\left(u_{k+1/2} - k_0 - k_1\bar{v}_{k+1/2} - k_2\bar{v}^2_{k+1/2}\right) \\ \lambda_{k+1} = \lambda_k \\ \mu_{k+1} = \mu_k + h\left(1 - \bar{\lambda}_{k+1/2} + \bar{\mu}_{k+1/2}\left(k_1 + 2\bar{v}_{k+1/2}k_2\right)\right) \\ s_0 = 0, \quad v_0 = 0, \quad v_N = 0, \quad \lambda_N = 0, \end{cases}$$

with $k = 0, \ldots, N-1$.
Control is explicit but *discontinuous*

$$u_{k+1/2} = -\text{sign}\left(\bar{\mu}_{k+1/2}\right)\left(g + k_3\bar{v}^2_{1/2}\right)$$

in this form the problem is hard to solve/approximate.

Demo example with PMP
Solution with Finite Difference 3/3

The unknowns of the problem are collected in $\boldsymbol{z} \in \mathbb{R}^{4N}$:

$$\boldsymbol{z} = (x_0, \dots, x_N, v_0, \dots, v_N, \lambda_0, \dots, \lambda_N, \mu_0, \dots, \mu_N)^T$$

The nonlinear system contains discontinuous function $\mathrm{sign}(x)$ which make it hard or impossible to solve. A working strategy is to smooth the Hamiltonian minimization with interior point approach

$$u^\star \approx \arg\min_{u \in \mathbb{R}} \Big(H(x, v, \lambda_1, \lambda_2, u) + b(u) \Big)$$

where $b(u)$ is a barrier function:

$$b(u) = -\varepsilon (g + k_3 v^2) \log \cos \left(\frac{\pi}{2} \frac{u}{g + k_3 v^2} \right)$$

and the minima satisfy $\frac{\partial}{\partial u} \Big(H(x, v, \lambda, \mu, u) + b(u) \Big) = 0$, so that control u as a function of μ and v can be easily computed

$$u(\mu, v) = -\frac{2}{\pi} \left(g + k_3 v^2 \right) \arctan \left(\frac{2\lambda_2}{\pi \epsilon} \right)$$

Demo example with PMP

Discrete Solution

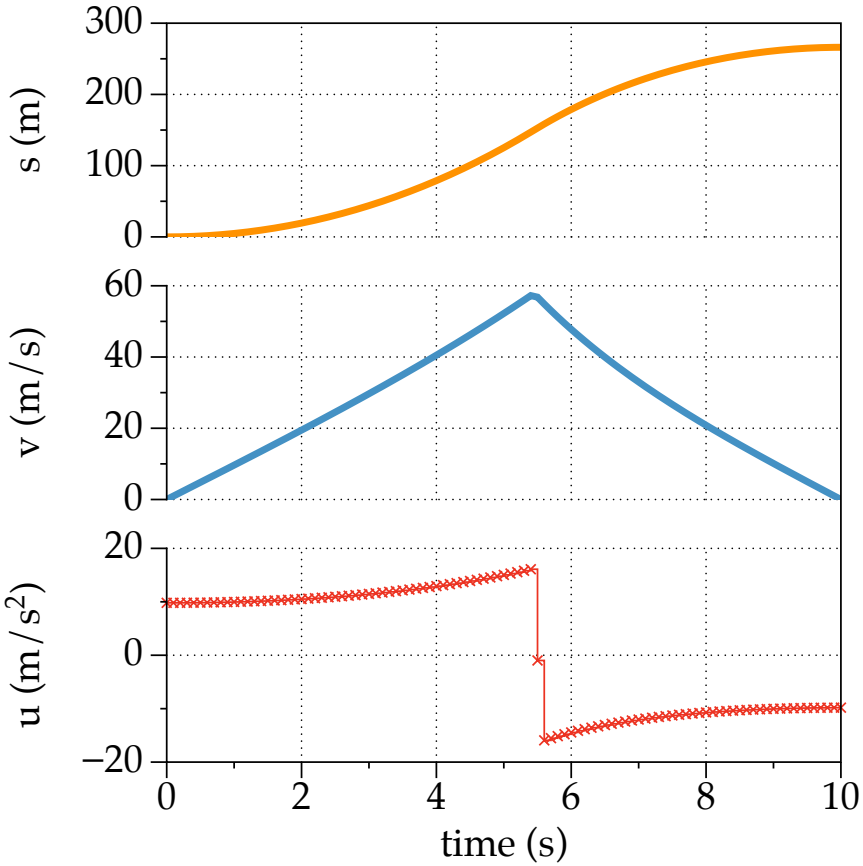


Figure: Solution with 100 intervals

Demo example with PMP

Solution comparison

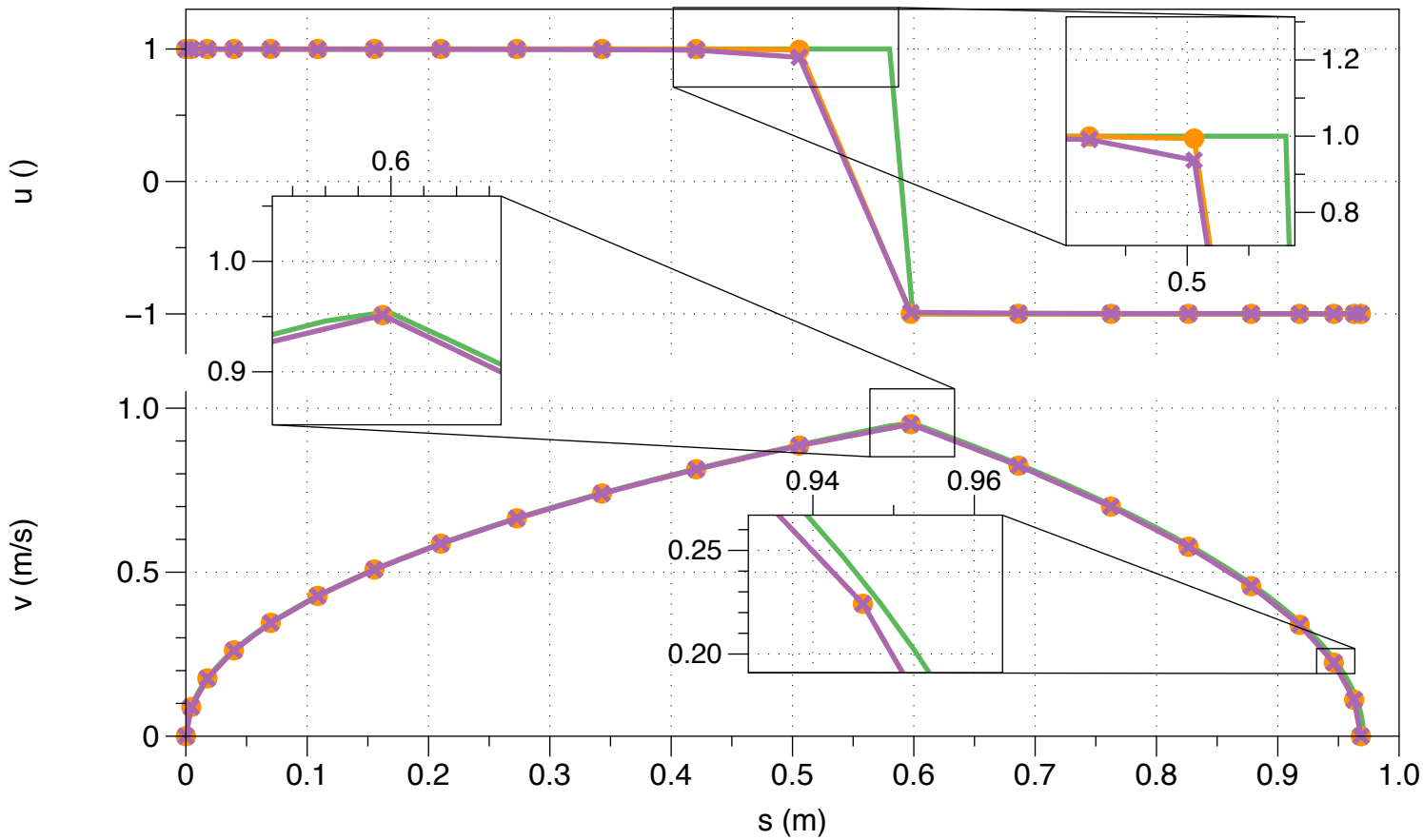


Figure: Solution comparison vs exact solution sampled

Demo example with PMP
Discussion: numerical issues

Method	Solver	cpu time (s) $N = 100$	cpu time (s) $N = 1000$	cpu time (s) $N = 1000$
Direct	IPOPT	0.08	0.6	≈8
Indirect (u-solved)	lsqnonlin	6.1	—	—
Indirect (u-solved)	STRSCNE	0.16	1.2	≈15
Indirect (u-solved)	newton	0.19	1.1	≈11
Indirect (u-solved)	PINS	0.05	0.2	0.85

Table: Computational time among different solution method and solver adopted. Newton solver is a simplified version of HYNESS solver as pure Matlab code. STRSCNE is a Scaled Trust-Region Solver for Constrained Nonlinear Equations as pure Matlab code. PINS is C++ indirect solver. `lsqnonlin()` do not converge for $N = 1000$, $N = 10000$.

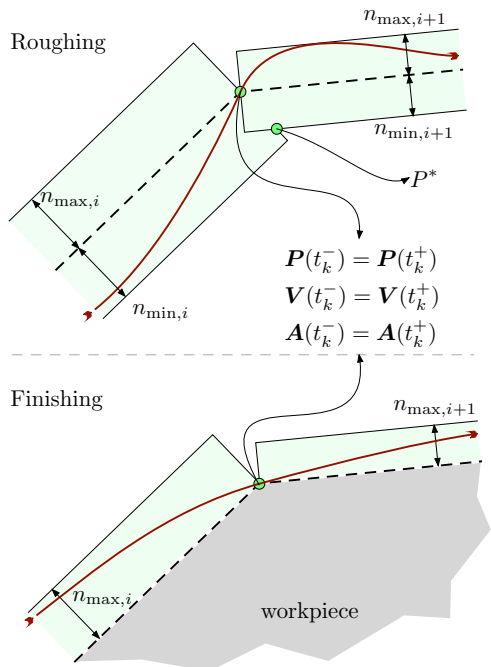
CN-Optimal Control: CNOC

Brief summary

The whole tool trajectory is optimized, and not only the feed-rate profile.
Lateral tracking tolerance on the tool-path can be defined (physically related to the workpiece design tolerance).

$$\min_{j_s, j_n} \int_0^{t_f} \left(\frac{\sqrt{v_s(t)^2 + v_n(t)^2}}{f(s(t))} - 1 \right)^2 \mathrm{d}t,$$

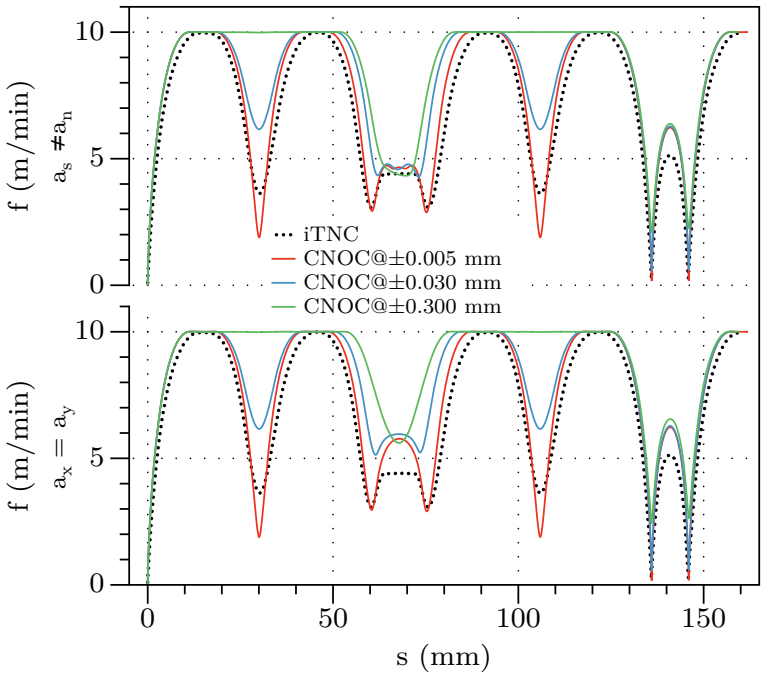
subject to a set of constraints on the dynamics of physical actuators (limits in speed, acceleration, and jerk) and on the maximum allowed tracking error.

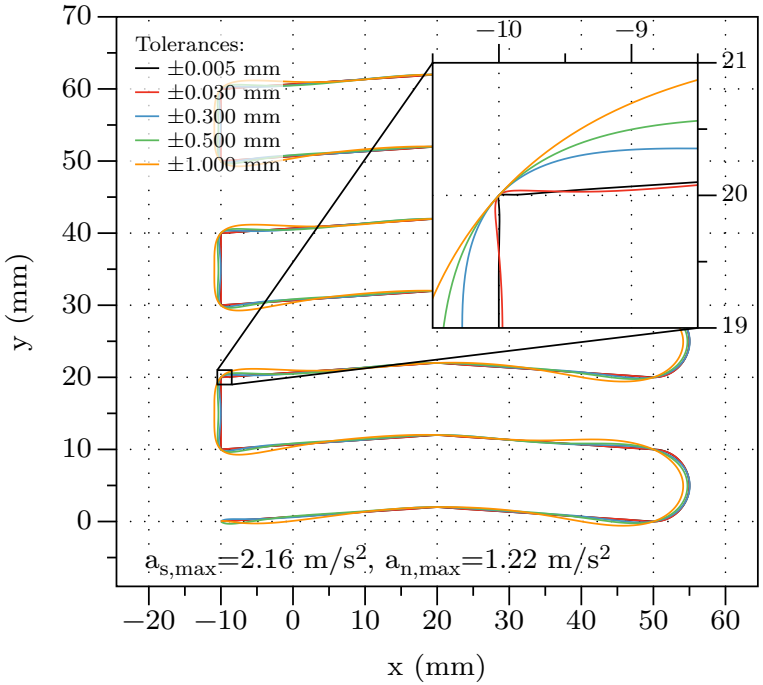


CN-Optimal Control: CNOC

Brief summary

Problem solved faster than realtime with free formulation geometrical space domain.
Speed (feed rate) can go to zero when necessary





Minimum Lap Time Application

Brief summary

Minimum lap time problem but with human-like manoeuvre:

$$\min_{S_r, S_f, \tau} \int_0^L \frac{1}{v_s(\zeta)} \mathrm{d}\zeta$$

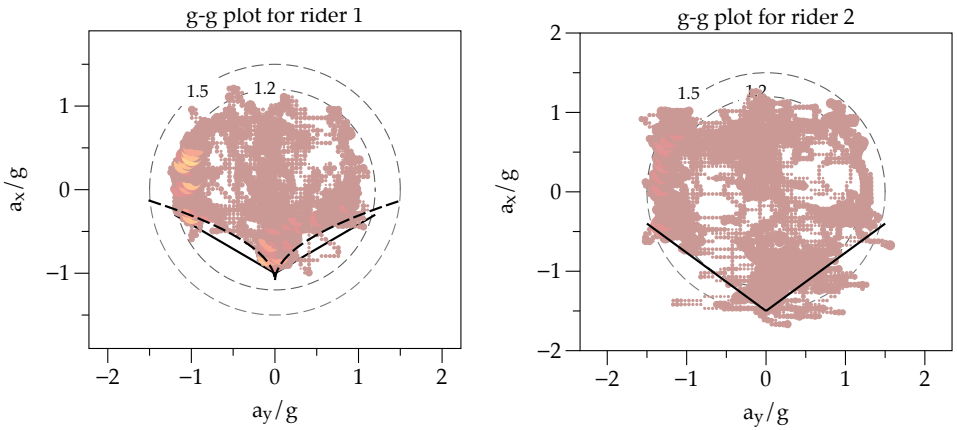
subject to:

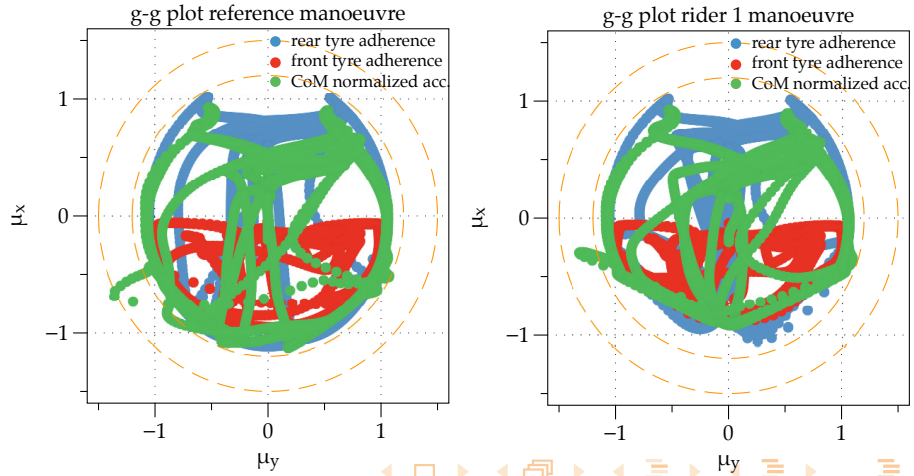
$$\boldsymbol{A}(\zeta)\boldsymbol{x}' = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$$

$$a_x(\zeta) \geq -a_{y_0} - \left|\frac{a_y(\zeta))}{a_{y_L}}\right|^{n_B}$$

$$\boldsymbol{c}(\boldsymbol{x}, \boldsymbol{u}) \geq 0$$

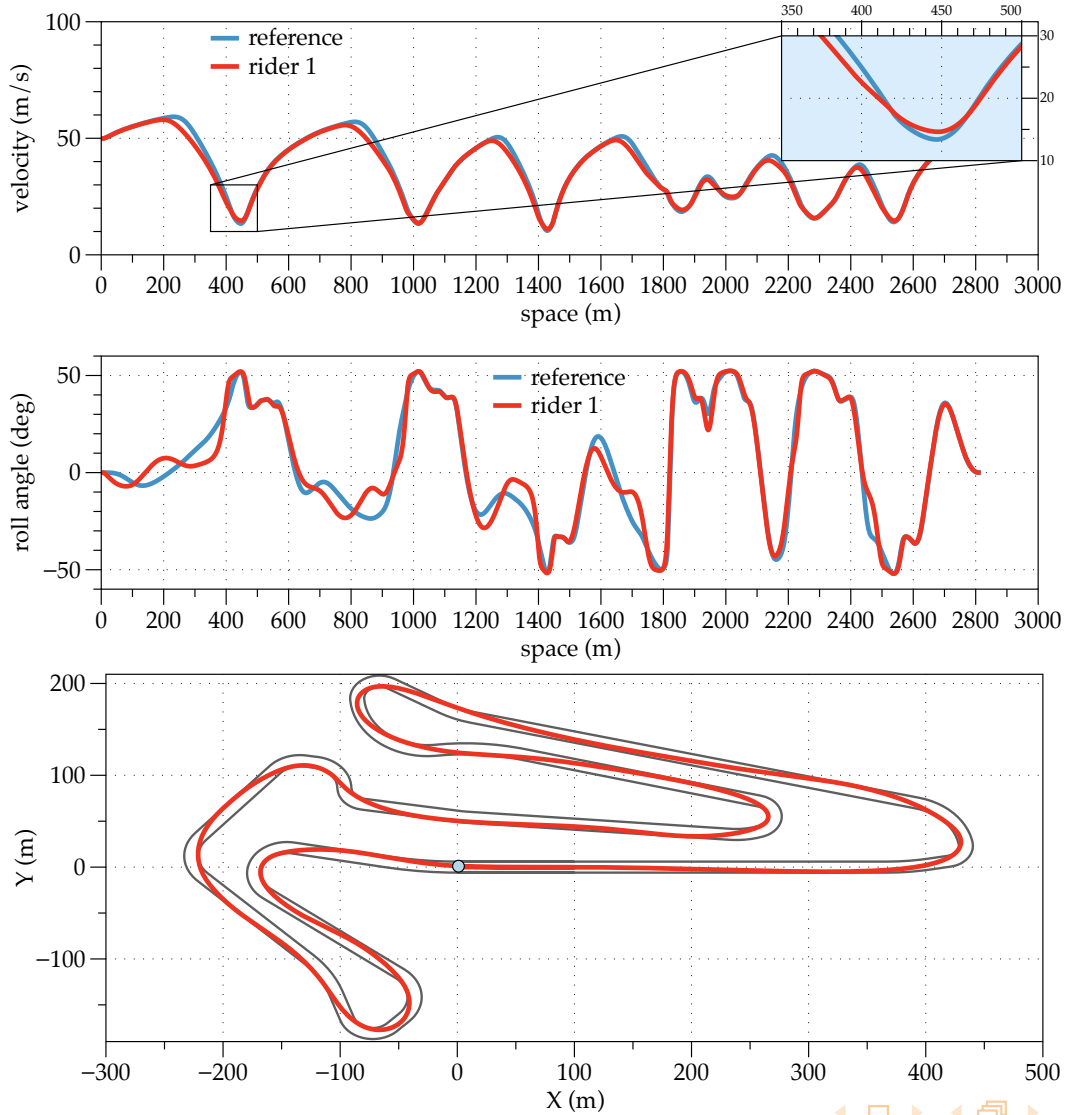
a_{y_0} and n_B are parameters that define the shape of the envelope.





Minimum Lap Time Application

Brief summary



Conclusion

NLP + SQP is easier to implement, is fast for medium problem but PMP can be still employed with some advantages:

- the PMP provides more accurate solutions
- the PMP allows to derive symbolic/approximated symbolic explicit optimal control laws
- FD approximation of BVP, with penalty functions and robust solver can be faster than NLP.
- the PMP allows to derive symbolic optimal control solutions
- PMP+FD approximation can handle very large problems.