Numerical Optimal Control

Moritz Diehl

Systems Control and Optimization Laboratory, University of Freiburg, Germany

Summer School on Robust Model Predictive Control with CasADi, University of Freiburg
September 15-19, 2025

(slides jointly developed with Armin Nurkanović)

universität freiburg

Continuous-Time Optimal Control Problems (OCP)

Continuous-Time OCP with Ordinary Differential Equation (ODE) Constraints

$$
\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & \int_0^T L_{\mathrm{c}}(x(t),u(t))\,\mathrm{d}t + E(x(T)) \\
\text{s.t.} \quad & x(0) = \bar{x}_0 \\
& \dot{x}(t) = f(x(t),u(t)) \\
& 0 \geq h(x(t),u(t)),\ t \in [0,T] \\
& 0 \geq r(x(T))
\end{aligned}
$$

Can in most applications assume convexity of all "outer" problem functions: L_{c}, E, h, r.

(More general optimal control problems)

Many features left out here for simplicity of presentation:

- multiple dynamic stages
- differential algebraic equations (DAE) instead of ODE
- explicit time dependence
- constant design parameters
- multipoint constraints $r(x(t_0), x(t_1), \ldots, x(t_{\mathrm{end}})) = 0$

Three Levels of Difficulty in Continuous-Time OCP

Continuous-Time OCP

$$\min_{x(\cdot),u(\cdot)} \int_0^T L_c(x(t),u(t))\,\mathrm{d}t + E(x(T))$$

$$\text{s.t.} \quad x(0) = \bar{x}_0$$

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

$$0 \geq h(x(t),u(t)),\ t \in [0,T]$$

$$0 \geq r(x(T))$$

Three levels of difficulty:

(a) Linear ODE: $f(x,u) = Ax + Bu$ ($\rightarrow$ convex optimization)

(b) Nonlinear smooth ODE: $f \in \mathcal{C}^1$ ($\rightarrow$ nonlinear optimization)

(c) Nonsmooth and Mixed-Integer Dynamics

In this school, we focus on cases (a) and (b).

Recall: Runge-Kutta Discretization for Ordinary Differential Equations

Ordinary Differential Equation (ODE)

$$\dot{x}(t) = \underbrace{f(x(t), u(t))}_{=:v(t)}$$

Initial Value Problem (IVP)

$$\begin{aligned} x(0) &= \bar{x}_0 \\ v(t) &= f(x(t), u(t)) \\ \dot{x}(t) &= v(t) \\ t &\in [0, T] \end{aligned}$$

Discretization: N Runge-Kutta steps of each n_s stages

$$\begin{aligned} x_{0,0} &= \bar{x}_0, & \Delta t &= \tfrac{T}{N} \\ v_{k,j} &= f(x_{k,j}, u_k) \\ x_{k,j} &= x_{k,0} + \Delta t \textstyle\sum_{n=1}^{n_s} a_{jn} v_{k,n} \\ x_{k+1,0} &= x_{k,0} + \Delta t \textstyle\sum_{n=1}^{n_s} b_n v_{k,n} \\ & j = 1, \ldots, n_s, \quad k = 0, \ldots, N-1 \end{aligned}$$

For fixed controls and initial value: square system with $n_x + N(2n_s + 1)n_x$ unknowns, implicitly defined via $n_x + N(2n_s + 1)n_x$ equations.
(trivial eliminations in case of explicit RK methods)

Direct Methods Transform OCP into Nonlinear Program (NLP)

Continuous time OCP

$$
\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & \int_0^T L_{\mathrm{c}}(x(t),u(t))\,\mathrm{d}t + E(x(T)) \\
\text{s.t.} \quad & x(0) = \bar{x}_0 \\
& \dot{x}(t) = f(x(t),u(t)) \\
& 0 \geq h(x(t),u(t)),\ t \in [0,T] \\
& 0 \geq r(x(T))
\end{aligned}
$$

- Direct methods "first discretize, then optimize"

Direct Methods Transform OCP into Nonlinear Program (NLP)

Continuous time OCP

$$\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & \int_0^T L_c(x(t),u(t))\,\mathrm{d}t + E(x(T)) \\
\text{s.t.} \quad & x(0) = \bar{x}_0 \\
& \dot{x}(t) = f(x(t),u(t)) \\
& 0 \geq h(x(t),u(t)),\ t \in [0,T] \\
& 0 \geq r(x(T))
\end{aligned}$$

- Direct methods "first discretize, then optimize"

1. Parameterize controls, e.g. $u(t) = u_n, t \in [t_n, t_{n+1}]$.

Direct Methods Transform OCP into Nonlinear Program (NLP)

Continuous time OCP

$$\min_{x(\cdot),u(\cdot)} \int_0^T L_{\mathrm{c}}(x(t),u(t))\,\mathrm{d}t + E(x(T))$$
$$\text{s.t.} \quad x(0) = \bar{x}_0$$
$$\dot{x}(t) = f(x(t),u(t))$$
$$0 \geq h(x(t),u(t)),\ t \in [0,T]$$
$$0 \geq r(x(T))$$

- Direct methods "first discretize, then optimize"

1. Parameterize controls, e.g. $u(t) = u_n, t \in [t_n, t_{n+1}]$.
2. Discretize cost and dynamics

$$L_{\mathrm{d}}(x_n, z_k, u_n) \approx \int_{t_n}^{t_{n+1}} L_{\mathrm{c}}(x(t),u(t))\,\mathrm{d}t$$

Replace $\dot{x} = f(x,u)$ by

$$x_{n+1} = \phi_f(x_n, z_n, u_n)$$
$$0 = \phi_{\mathrm{int}}(x_n, z_n, u_n)$$

Direct Methods Transform OCP into Nonlinear Program (NLP)

Continuous time OCP

$$\begin{aligned}\min_{x(\cdot),u(\cdot)} \quad & \int_0^T L_{\mathrm{c}}(x(t),u(t))\,\mathrm{d}t + E(x(T)) \\ \text{s.t.} \quad & x(0) = \bar{x}_0 \\ & \dot{x}(t) = f(x(t),u(t)) \\ & 0 \geq h(x(t),u(t)),\ t \in [0,T] \\ & 0 \geq r(x(T))\end{aligned}$$

- Direct methods "first discretize, then optimize"

1. Parameterize controls, e.g. $u(t) = u_n, t \in [t_n, t_{n+1}]$.
2. Discretize cost and dynamics

$$L_{\mathrm{d}}(x_n, z_k, u_n) \approx \int_{t_n}^{t_{n+1}} L_{\mathrm{c}}(x(t),u(t))\,\mathrm{d}t$$

Replace $\dot{x} = f(x,u)$ by

$$x_{n+1} = \phi_f(x_n, z_n, u_n)$$
$$0 = \phi_{\mathrm{int}}(x_n, z_n, u_n)$$

3. Also discretize path constraints

$$0 \geq \phi_h(x_n, z_n, u_n),\ n = 0, \ldots N-1.$$

Direct Methods Transform OCP into Nonlinear Program (NLP)

Continuous time OCP

$$\min_{x(\cdot),u(\cdot)} \int_0^T L_c(x(t),u(t))\,\mathrm{d}t + E(x(T))$$

$$\begin{aligned} \text{s.t.} \quad x(0) &= \bar{x}_0 \\ \dot{x}(t) &= f(x(t),u(t)) \\ 0 &\geq h(x(t),u(t)),\ t\in[0,T] \\ 0 &\geq r(x(T)) \end{aligned}$$

- Direct methods "first discretize, then optimize"

Discrete time OCP (an NLP)

$$\min_{\mathbf{x},\mathbf{z},\mathbf{u}} \sum_{k=0}^{N-1} L_{\mathrm{d}}(x_k,z_k,u_k) + E(x_N)$$

$$\begin{aligned} \text{s.t.} \quad x_0 &= \bar{x}_0 \\ x_{n+1} &= \phi_f(x_n,z_n,u_n) \\ 0 &= \phi_{\mathrm{int}}(x_n,z_n,u_n) \\ 0 &\geq \phi_h(x_n,z_n,u_n),\ n=0,\dots,N-1 \\ 0 &\geq r(x_N) \end{aligned}$$

Variables $\mathbf{x}=(x_0,\dots,x_N)$, $\mathbf{z}=(z_0,\dots,z_N)$ and $\mathbf{u}=(u_0,\dots,u_{N-1})$.
Here, $\mathbf{z}$ are the intermediate variables of the integrator (e.g. Runge-Kutta)

Simplest Direct Transcription: Single Step Explicit Euler

(not recommended in practice, other Runge-Kutta methods are much more efficient)

Continuous time OCP

$$\min_{x(\cdot),u(\cdot)} \int_0^T L_c(x(t),u(t))\,\mathrm{d}t + E(x(T))$$

$$\begin{aligned} \text{s.t.} \quad & x(0) = \bar{x}_0 \\ & \dot{x}(t) = f(x(t),u(t)) \\ & 0 \geq h(x(t),u(t)),\ t \in [0,T] \\ & 0 \geq r(x(T)) \end{aligned}$$

- Direct methods: first discretize, then optimize

Single Step Explicit Euler NLP, with $\Delta t = \frac{T}{N}$

$$\min_{\mathbf{x},\mathbf{u}} \sum_{k=0}^{N-1} L_c(x_k,u_k)\Delta t + E(x_N)$$

$$\begin{aligned} \text{s.t.} \quad & x_0 = \bar{x}_0 \\ & x_{n+1} = x_n + f(x_n,u_n)\Delta t \\ & 0 \geq h(x_n,u_n),\ n = 0,\dots,N-1 \\ & 0 \geq r(x_N) \end{aligned}$$

Variables $\mathbf{x} = (x_0,\dots,x_N)$ and $\mathbf{u} = (u_0,\dots,u_{N-1})$.
(single step explicit Euler has no internal integrator variables $\mathbf{z}$)

Sparse NLP resulting from direct transcription

Discrete time OCP (an NLP)

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{z},\mathbf{u}} \ & \textstyle\sum_{k=0}^{N-1} L_{\mathrm{d}}(x_k, z_n, u_k) + E(x_N) \\
\text{s.t.} \quad & x_0 = \bar{x}_0 \\
& x_{n+1} = \phi_f(x_n, z_n, u_n) \\
& 0 = \phi_{\mathrm{int}}(x_n, z_n, u_n) \\
& 0 \geq \phi_h(x_n, z_n, u_n), \ n = 0, \ldots, N-1 \\
& 0 \geq r(x_N)
\end{aligned}
$$

Variables $w = (\mathbf{x}, \mathbf{z}, \mathbf{u})$

Sparse NLP resulting from direct transcription

Discrete time OCP (an NLP)

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{z},\mathbf{u}} \ & \textstyle\sum_{k=0}^{N-1} L_{\mathrm{d}}(x_k, z_n, u_k) + E(x_N) \\
\text{s.t.} \quad & x_0 = \bar{x}_0 \\
& x_{n+1} = \phi_f(x_n, z_n, u_n) \\
& 0 = \phi_{\mathrm{int}}(x_n, z_n, u_n) \\
& 0 \geq \phi_h(x_n, z_n, u_n), \ n = 0, \ldots, N-1 \\
& 0 \geq r(x_N)
\end{aligned}
$$

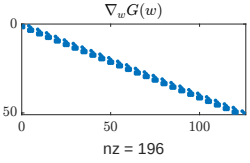
Variables $w = (\mathbf{x}, \mathbf{z}, \mathbf{u})$

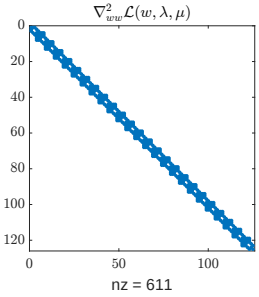
Nonlinear Program (NLP)

$$
\begin{aligned}
\min_{w \in \mathbb{R}^{n_x}} \ & F(w) \\
\text{s.t.} \ & G(w) = 0 \\
& H(w) \geq 0
\end{aligned}
$$

Large and sparse NLP

Sparse NLP resulting from direct transcription





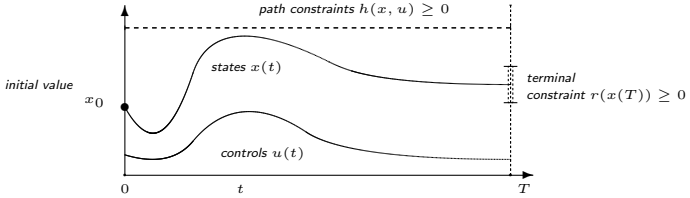
Variables $w = (\mathbf{x}, \mathbf{z}, \mathbf{u})$

Nonlinear Program (NLP)

$$\begin{aligned} \min_{w \in \mathbb{R}^{n_x}} \quad & F(w) \\ \text{s.t.} \quad & G(w) = 0 \\ & H(w) \geq 0 \end{aligned}$$

Large and sparse NLP

Simplified Optimal Control Problem in ODE



Continuous Time Optimal Control Problem

$$\begin{aligned}
&\underset{x(\cdot),u(\cdot)}{\text{minimize}} \quad \int_0^T L(x(t),u(t))\,dt \;+\; E\left(x(T)\right) \\
&\text{subject to} \\
&x(0) - x_0 = 0, && \text{(fixed initial value)} \\
&\dot{x}(t) - f(x(t),u(t)) = 0, \quad t \in [0,T], && \text{(ODE model)} \\
&h(x(t),u(t)) \geq 0, \quad t \in [0,T], && \text{(path constraints)} \\
&r\left(x(T)\right) \geq 0 && \text{(terminal constraints)}
\end{aligned}$$

Direct Methods

- "first discretize, then optimize"
- transcribe infinite problem into finite **Nonlinear Programming Problem (NLP)**
- Pros and Cons:
  - \+ can use state-of-the-art methods for NLP solution
  - \+ can treat inequality constraints and multipoint constraints much easier
  - \- obtains only suboptimal / approximate solution
- nowadays most commonly used methods due to their easy applicability and robustness

Classification of Direct Optimal Control Methods

Direct methods transform continuous time problem into a nonlinear program (NLP):

- Direct Transcription: all internal integrator variables are kept exposed as NLP variables. Special cases: direct collocation and pseudospectral methods. (called "simultaneous approach", as simulation and optimization are tackled simultaneously by NLP solver)
- Direct Multiple Shooting: for every control interval, all internal integration steps are hidden to the NLP. Integration routine is complicated but differentiable function (also called "simultaneous approach")
- Direct Single Shooting: all state variables are eliminated by forward simulation, only the control parameters are kept as NLP variables. NLP objective and constraints are very long functions. (called "sequential approach", as simulation and optimization proceed sequentially)
- Flatness-based optimal control: in "flat" systems, the states and control inputs can be obtained from derivatives of a "flat output". One can then parameterize the flat output as superposition of smooth basis functions, and formulate an NLP in the space of the basis coefficients. Similar in performance to simultaneous approaches but limited to flat systems.

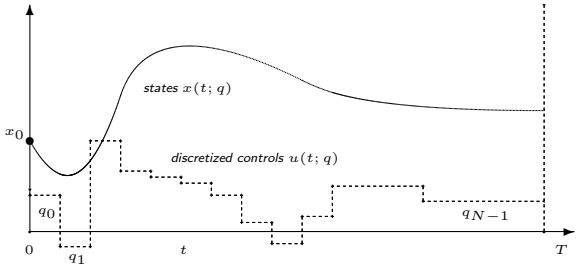
Direct Methods: Comparison of Sequential and Simultaneous Approach

We compare two direct methods:

- Direct Single Shooting (sequential simulation and optimization)
- Direct Multiple Shooting (simultaneous simulation and optimization)

Direct Single Shooting [Hicks1971,Sargent1978]

Discretize controls $u(t)$ on fixed grid $0 = t_0 < t_1 < \ldots < t_N = T$, regard states $x(t)$ on $[0, T]$ as dependent variables.



Use numerical integration to obtain state as function $x(t; q)$ of finitely many control parameters $q = (q_0, q_1, \ldots, q_{N-1}) \in \mathbb{R}^{N \cdot n_u}$

NLP in Direct Single Shooting

After control discretization and numerical ODE solution, obtain NLP:

NLP resulting from Direct Single Shooting

$$
\begin{aligned}
&\underset{q \in \mathbb{R}^{N \cdot n_u}}{\text{minimize}} && \int_0^T L(x(t;q), u(t;q))\, dt + E\left(x(T;q)\right) \\
&\text{subject to} \\
&\quad h(x(t_i;q), u(t_i;q)) \geq 0, \quad i = 0, \ldots, N, && \textit{(discretized path constraints)} \\
&\quad r\left(x(T;q)\right) \geq 0. && \textit{(terminal constraints)}
\end{aligned}
$$

Solve with nonlinear programming solver, e.g. Sequential Quadratic Programming (SQP)

Solution by Standard SQP

Summarize problem as $\boxed{\min_q F(q) \;\text{ s.t. }\; H(q) \geq 0}$

Solve e.g. by Sequential Quadratic Programming (SQP), starting with guess q^0 for controls. $k := 0$

1. Evaluate $F(q^k), H(q^k)$ by ODE solution, and derivatives
2. Compute correction Δq^k by solution of QP:

$$\boxed{\min_{\Delta q} \nabla F(q_k)^T \Delta q + \tfrac{1}{2} \Delta q^T A^k \Delta q \;\text{ s.t. }\; H(q^k) + \nabla H(q^k)^T \Delta q \geq 0}$$

3. Perform step $q^{k+1} = q^k + \alpha_k \Delta q^k$ with step length $\alpha_k \in (0, 1]$ determined by line search

ODE Sensitivities

How to compute the sensitivity $\frac{\partial x(t;q)}{\partial q}$ of a numerical ODE solution $x(t;q)$ with respect to the controls q?

many ways, for example:

- External Numerical Differentiation (END)
- Variational Differential Equations
- Automatic Differentiation (AD) of integration code
- Internal Numerical Differentiation (IND)

cf. [Rien Quirynen, Numerical simulation methods for embedded optimization, PhD thesis, KU Leuven and Freiburg University, 2017]

Numerical Test Problem

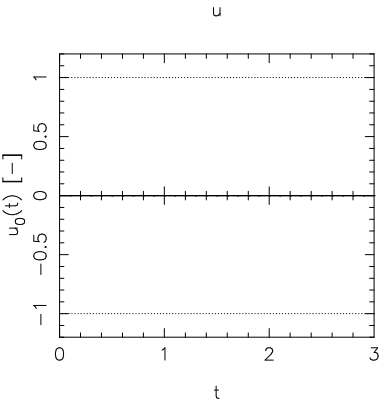
$$
\begin{aligned}
&\underset{x(\cdot),u(\cdot)}{\text{minimize}} \quad \int_0^3 x(t)^2 + u(t)^2 \, dt \\
&\text{subject to} \\
&\qquad x(0) = x_0, && \text{(initial value)} \\
&\qquad \dot{x} = (1+x)x + u, \quad t \in [0,3], && \text{(ODE model)} \\
&\begin{bmatrix} 1 - x(t) \\ 1 + x(t) \\ 1 - u(t) \\ 1 + u(t) \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad t \in [0,3], && \text{(bounds)} \\
&\qquad x(3) = 0. && \text{(zero terminal constraint)}
\end{aligned}
$$

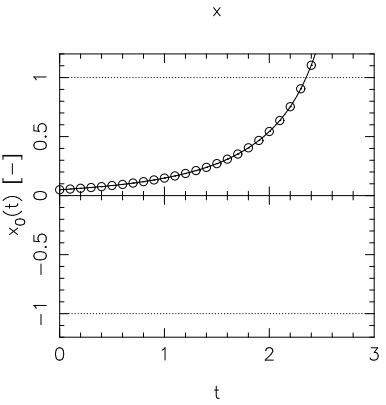
Remark: Uncontrollable growth for $(1 + x_0)x_0 - 1 \geq 0 \Leftrightarrow x_0 \geq 0.618$.

Single Shooting Optimization for $x_0 = 0.05$

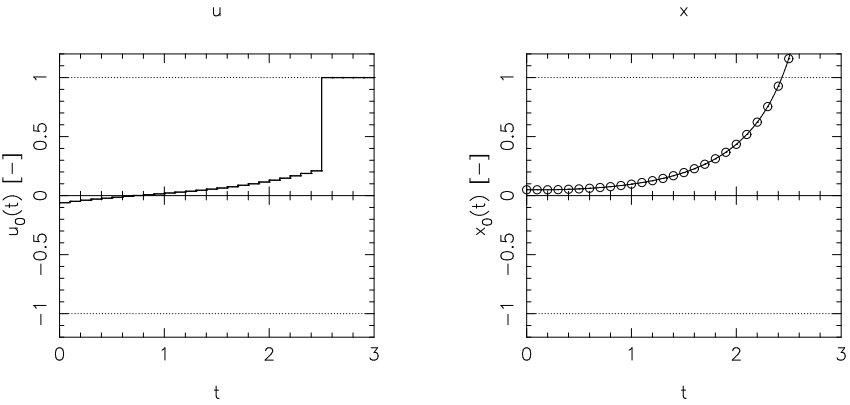
- choose $N = 30$ equal control intervals
- initialize with steady state controls $u(t) \equiv 0$
- initial value $x_0 = 0.05$ is the maximum possible for the problem to be solved by single shooting, because the initial trajectory explodes otherwise

Single Shooting: Initialization

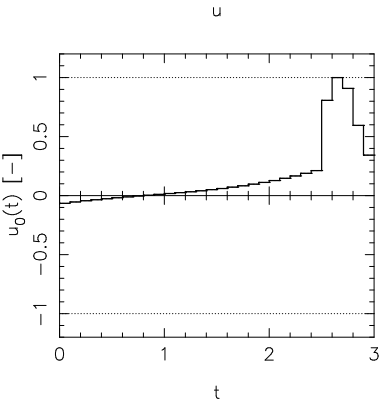


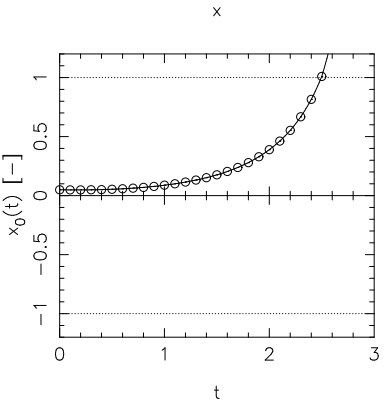


Single Shooting: First Iteration

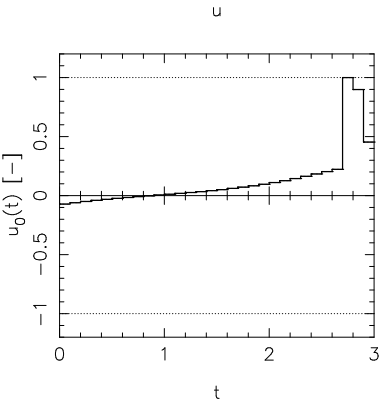


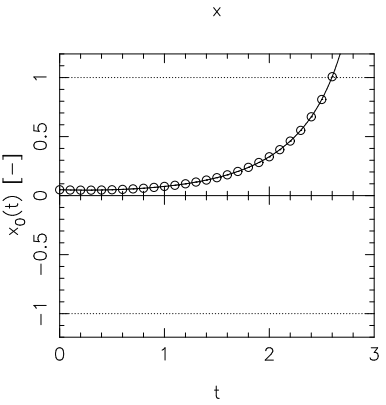
Single Shooting: 2nd Iteration



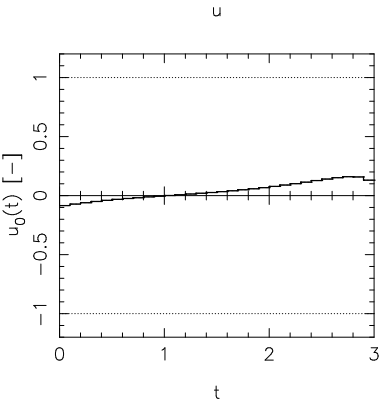


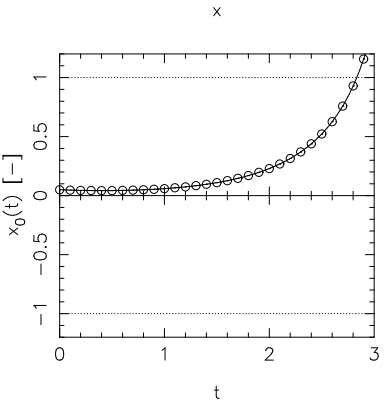
Single Shooting: 3rd Iteration



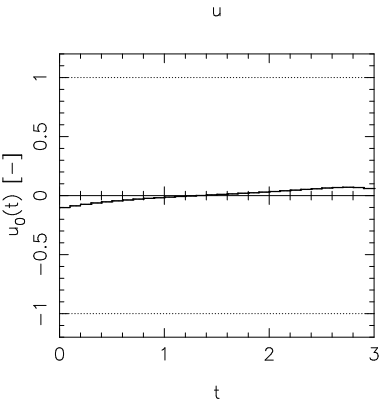


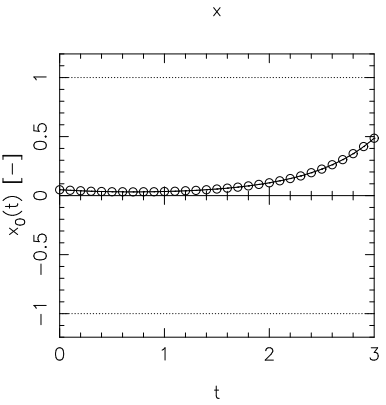
Single Shooting: 4th Iteration



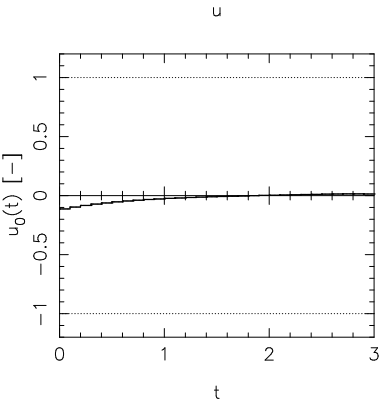


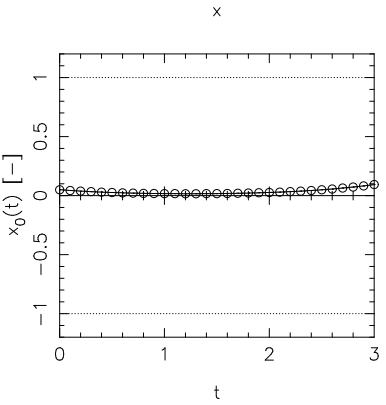
Single Shooting: 5th Iteration



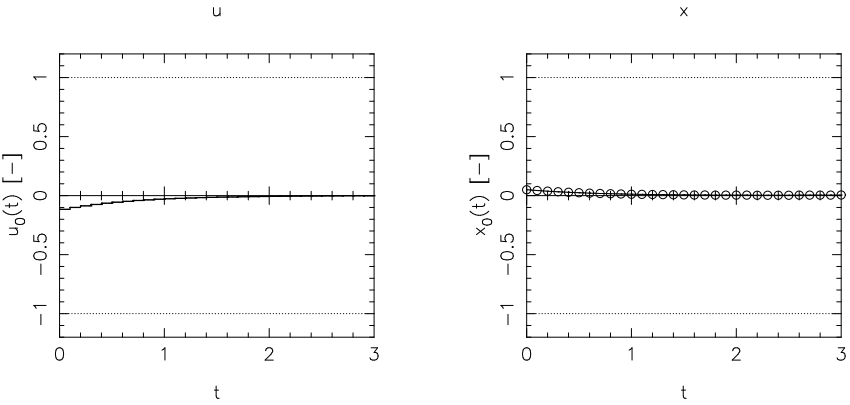


Single Shooting: 6th Iteration





Single Shooting: 7th Iteration and Solution



Direct Single Shooting: Pros and Cons

- **sequential** simulation and optimization.
- \+ can use state-of-the-art ODE/DAE solvers
- \+ few degrees of freedom even for large ODE/DAE systems
- \+ active set changes easily treated
- \+ need only initial guess for controls q
- \- cannot use knowledge of x in initialization (e.g. in tracking problems)
- \- ODE solution $x(t;q)$ can depend very nonlinearly on q
- \- unstable systems difficult to treat
- often used in self-made optimal control codes in engineering applications

Direct Multiple Shooting [Bock 1984]

- Discretize controls piecewise on a coarse grid

$$u(t) = q_i \quad \text{for} \quad t \in [t_i, t_{i+1}]$$

- Solve ODE on each interval $[t_i, t_{i+1}]$ numerically, starting with artificial initial value s_i:

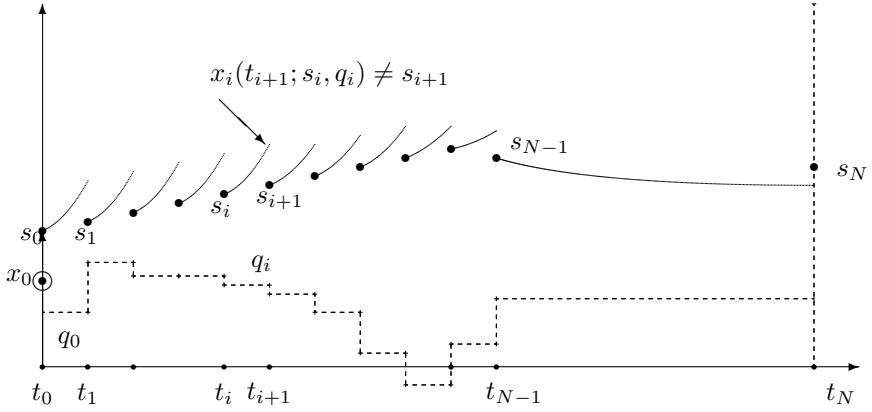
$$\begin{aligned} \dot{x}_i(t; s_i, q_i) &= f(x_i(t; s_i, q_i), q_i), \quad t \in [t_i, t_{i+1}], \\ x_i(t_i; s_i, q_i) &= s_i. \end{aligned}$$

  Obtain trajectory pieces $x_i(t; s_i, q_i)$.

- Also numerically compute integrals

$$l_i(s_i, q_i) := \int_{t_i}^{t_{i+1}} L(x_i(t_i; s_i, q_i), q_i) dt$$

Sketch of Direct Multiple Shooting



NLP in Direct Multiple Shooting

$$\underset{s,q}{\text{minimize}} \quad \sum_{i=0}^{N-1} l_i(s_i, q_i) \;+\; E\,(s_N)$$

subject to

$$\begin{aligned}
s_0 - x_0 &= 0, && \text{(initial value)}\\
s_{i+1} - x_i(t_{i+1}; s_i, q_i) &= 0,\ i = 0, \ldots, N-1, && \text{(continuity)}\\
h(s_i, q_i) &\geq 0,\ i = 0, \ldots, N, && \text{(discretized path constraints)}\\
r\,(s_N) &\geq 0. && \text{(terminal constraints)}
\end{aligned}$$

Multiple Shooting NLP = Discrete Time Optimal Control Problem

Discrete Time Optimal Control Problem

$$
\begin{aligned}
\min_{x,u} \quad & \sum_{k=0}^{N-1} L(x_k, u_k) + E(x_N) \\
\text{s.t.} \quad & x_0 = \bar{x}_0 \\
& x_{k+1} = f(x_k, u_k) \\
& h(x_k, u_k) \geq 0, \qquad k = 0, \ldots, N-1 \\
& r(x_N) \geq 0
\end{aligned}
$$

summarize all variables as $w := (s_0, q_0, s_1, q_1, \ldots, s_N)$

Nonlinear Program

$$
\begin{aligned}
\min_{w} \quad & F(w) \\
\text{s.t.} \quad & G(w) = 0 \\
& H(w) \geq 0
\end{aligned}
$$

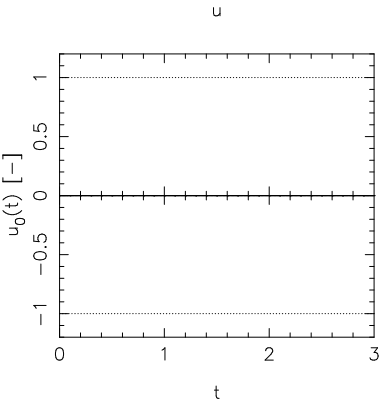
Structured Block-Sparse NLP

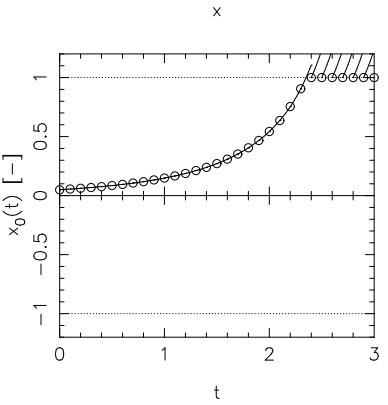
Nonlinear Program

$$\begin{aligned} \min_{w} \; & F(w) \\ \text{s.t.} \quad & G(w) = 0 \\ & H(w) \geq 0 \end{aligned}$$

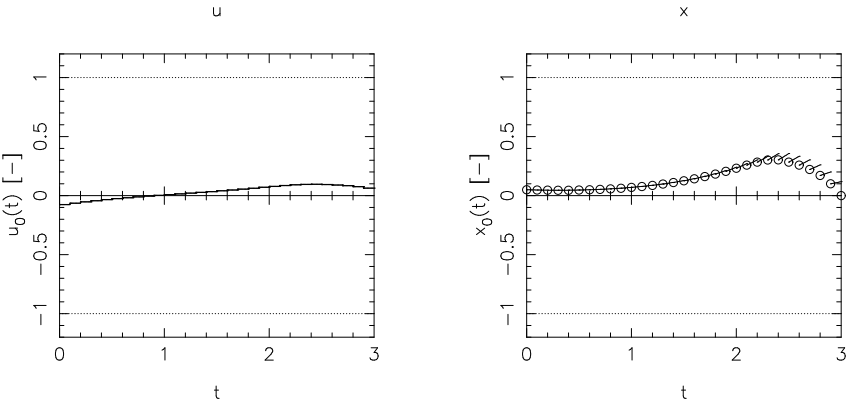
- Jacobian $\nabla G(w)^\top$ contains linearized dynamic model equations
- Jacobians and Hessian of NLP are block sparse, can be exploited in numerical solution
- NLPs of single and direct multiple shooting are equivalent (same solutions in control space)
- but "lifting" of the state variables of multiple shooting reduces the nonlinearity, as observed by many practitioners and investigated theoretically by [Albersmeyer and Diehl, The Lifted Newton Method and Its Application to Optimization, SIAM J. Opt., 2010]

Test Example: Initialization with $u(t) \equiv 0$

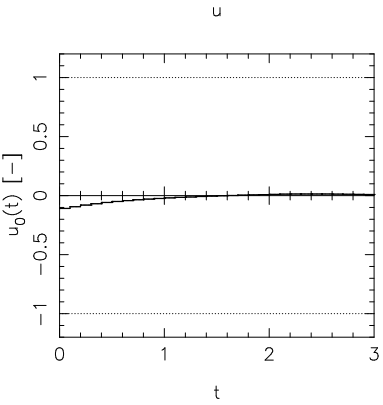


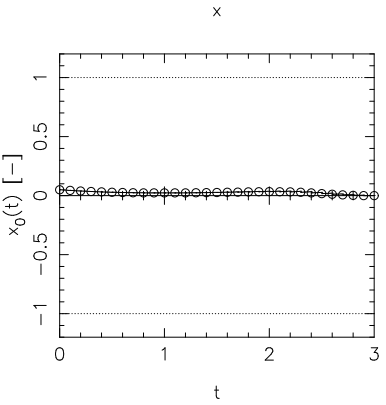


Multiple Shooting: First Iteration

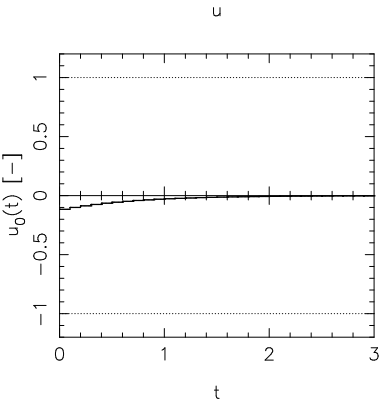


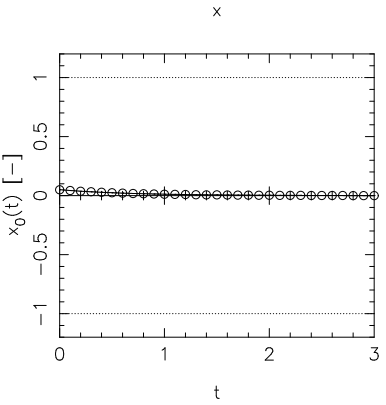
Multiple Shooting: 2nd Iteration





Multiple Shooting: 3rd Iteration and Solution





Direct Multiple Shooting: Pros and Cons

- **simultaneous** simulation and optimization.
- \+ uses **adaptive** ODE/DAE solvers
- \+ but NLP has **fixed dimensions**
- \+ can use knowledge of x in initialization (important in online context)
- \+ can treat unstable systems well
- \+ robust handling of path and terminal constraints
- \+ easy to parallelize
- \- not as sparse as collocation
- used for practical optimal control in many codes e.g MUSCOD (Bock), HQP (Franke), MUSCOD-II (Leineweber et al.), ACADO Toolkit (Houska, Ferreau et al.), acados (Verschueren, Frey, Frison, Kouzoupis, Quirynen et al.), ...

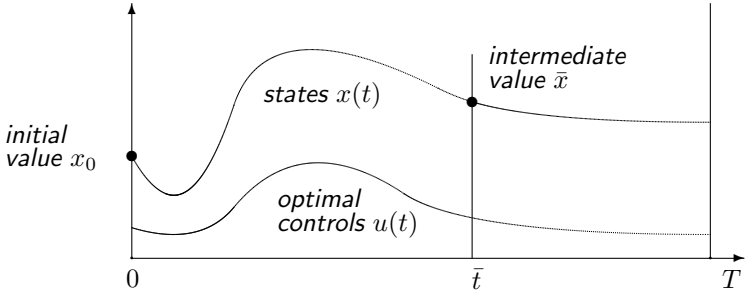
Dynamic Programming adresses Direct Multiple Shooting Problem

Discrete Time Optimal Control Problem

$$
\begin{aligned}
\min_{x,u} \quad & \sum_{k=0}^{N-1} L(x_k, u_k) + E(x_N) \\
\text{s.t.} \quad & x_0 = \bar{x}_0 \\
& x_{k+1} = f(x_k, u_k) \\
& h(x_k, u_k) \geq 0, \qquad k = 0, \ldots, N-1 \\
& r(x_N) \geq 0
\end{aligned}
$$

Principle of Optimality

Any subarc of an optimal trajectory is also optimal.



Subarc on $[\bar{t}, T]$ is optimal solution for initial value $\bar{x}$.

Dynamic Programming Cost-to-go (discrete time, unconstrained)

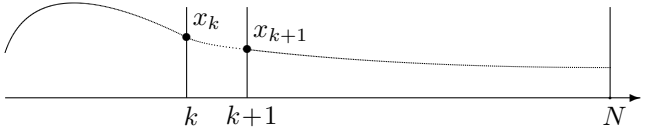
IDEA:

- Introduce **optimal-cost-to-go** function on $[k, N]$

$$J_k(x) := \min_{s_k, u_k, \ldots, s_N} \sum_{i=k}^{N-1} L(s_i, u_i) + E(s_N) \quad \text{s.t.} \quad s_k = x, \ldots$$

- Use **principle of optimality** on intervals $[k, k+1]$:

$$J_k(x_k) = \min_{s_k, u_k, s_{k+1}} L(s_k, a_k) + J_{k+1}(s_{k+1})$$
$$\text{s.t.} \quad s_k = x_k,\ s_{k+1} = f(s_k, u_k)$$



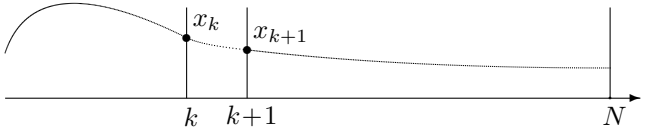
Dynamic Programming Step

Can simplify

$$J_k(x_k) = \min_{s_k, u_k, s_{k+1}} L(s_k, u_k) + J_{k+1}(s_{k+1})$$
$$\text{s.t.} \quad s_k = x_k,\ s_{k+1} = f(s_k, u_k)$$

by trivial elimination of s_k, s_{k+1} to

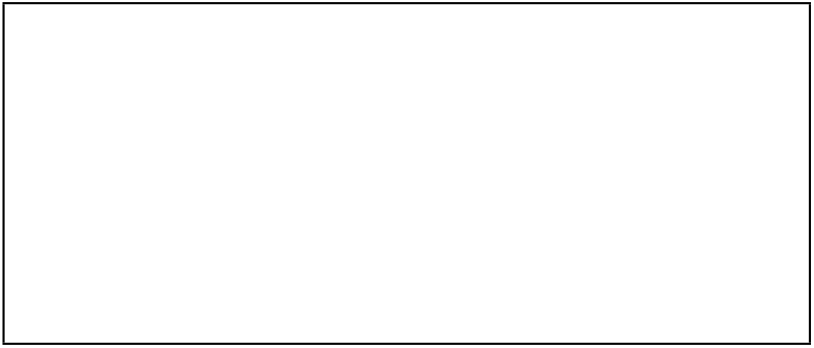
$$J_k(x_k) = \min_{u_k} L(x_k, u_k) + J_{k+1}(f(x_k, u_k))$$



Dynamic Programming Recursion

Iterate backwards, starting from $J_N(x) := E(x)$ for all $x \in \mathbb{R}^{n_x}$
for $k = N-1, N-2, \ldots$

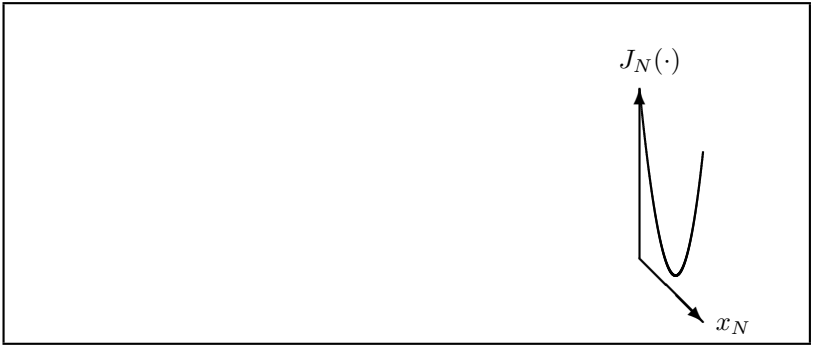
$$J_k(x) = \min_u L(x,u) + J_{k+1}(f(x,u))$$



Dynamic Programming Recursion

Iterate backwards, starting from $J_N(x) := E(x)$ for all $x \in \mathbb{R}^{n_x}$
for $k = N-1, N-2, \ldots$

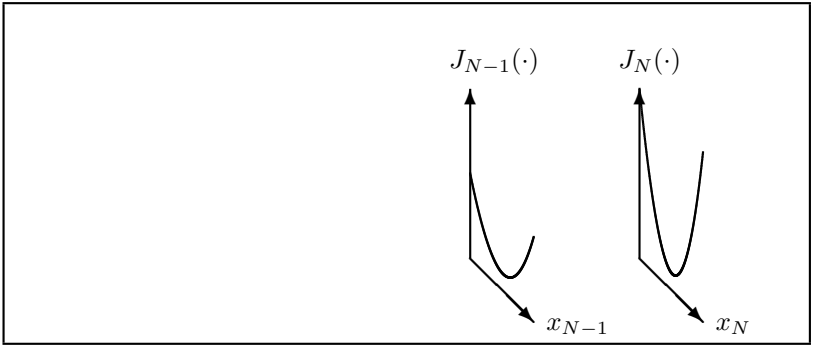
$$J_k(x) = \min_u L(x,u) + J_{k+1}(f(x,u))$$



Dynamic Programming Recursion

Iterate backwards, starting from $J_N(x) := E(x)$ for all $x \in \mathbb{R}^{n_x}$
for $k = N-1, N-2, \ldots$

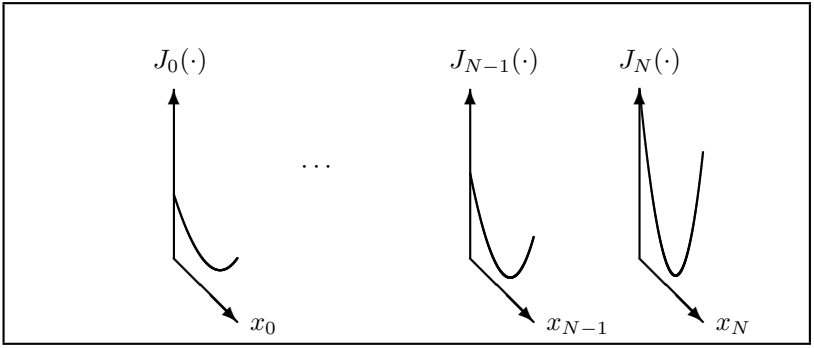
$$J_k(x) = \min_u L(x,u) + J_{k+1}(f(x,u))$$



Dynamic Programming Recursion

Iterate backwards, starting from $J_N(x) := E(x)$ for all $x \in \mathbb{R}^{n_x}$
for $k = N-1, N-2, \ldots$

$$J_k(x) = \min_u L(x,u) + J_{k+1}(f(x,u))$$



The optimal feedback control policy

The **optimal feedback control law** π_k^* at time k is defined by

$$\pi_k^*(x) \quad := \quad \arg\min_u \; L(x,u) + J_{k+1}(f(x,u))$$

These feedback laws together define the **optimal feedback control policy** $(\pi_0^*, \ldots, \pi_{N-1}^*)$ which tells us for any state x at any time index k what would be the optimal control action.

How to obtain optimal trajectories ?

The optimal policy $(\pi_0^*, \ldots, \pi_{N-1}^*)$ allows us to solve the original optimal control problem.

Starting with $x_0^* := \bar{x}_0$, we simulate the closed loop system for $k = 0, 1, \ldots, N-1$:

$$\begin{aligned} u_k^* &:= \pi_k^*(x_k^*) \\ x_{k+1}^* &:= f(x_k^*, u_k^*) \end{aligned}$$

yielding the optimal trajectories $x^* = (x_0^*, \ldots, x_N^*)$ and $u^* = (u_0^*, \ldots, u_N^*)$ that solve problem (2).

Optimal Control Problem

$$\begin{aligned} \min_{x,u} \quad & \sum_{k=0}^{N-1} L(x_k, u_k) + E(x_N) \\ \text{s.t.} \quad & x_0 = \bar{x}_0 \\ & x_{k+1} = f(x_k, u_k), \\ & \qquad k = 0, \ldots, N-1 \end{aligned} \tag{2}$$

How to obtain optimal trajectories ?

The optimal policy $(\pi_0^*, \ldots, \pi_{N-1}^*)$ allows us to solve the original optimal control problem.

Starting with $x_0^* := \bar{x}_0$, we simulate the closed loop system for $k = 0, 1, \ldots, N-1$:

$$\begin{aligned} u_k^* &:= \pi_k^*(x_k^*) \\ x_{k+1}^* &:= f(x_k^*, u_k^*) \end{aligned}$$

yielding the optimal trajectories $x^* = (x_0^*, \ldots, x_N^*)$ and $u^* = (u_0^*, \ldots, u_N^*)$ that solve problem (2).

Optimal Control Problem

$$\begin{aligned} \min_{x,u} \quad & \sum_{k=0}^{N-1} L(x_k, u_k) + E(x_N) \\ \text{s.t.} \quad & x_0 = \bar{x}_0 \\ & x_{k+1} = f(x_k, u_k), \\ & \qquad k = 0, \ldots, N-1 \end{aligned} \tag{2}$$

Note: MPC applies only $\pi_0^*(\bar{s}_0)$. The MPC law can be generated in one of three ways:

(a) via dynamic programming,
(b) via online solution of (2) in classical MPC, or
(c) via offline solution of (2) based on parametric programming in *explicit MPC*.

Robust Dynamic Programming

Dynamic Programming can straightforwardly be extended to games like chess, or to closed loop robust min-max optimal control problems, which are not easily treatable with other robust optimization methods.

Here, in each time step, we first choose the controls u_k, but then an adverse player choses disturbances w_k, and both influence the system dynamics $x_{k+1} = f(x_k, u_k, w_k)$.

Robust DP Recursion

Iterate backwards, from $k = N-1$ down to $k = 0$, using the robust Bellman equation

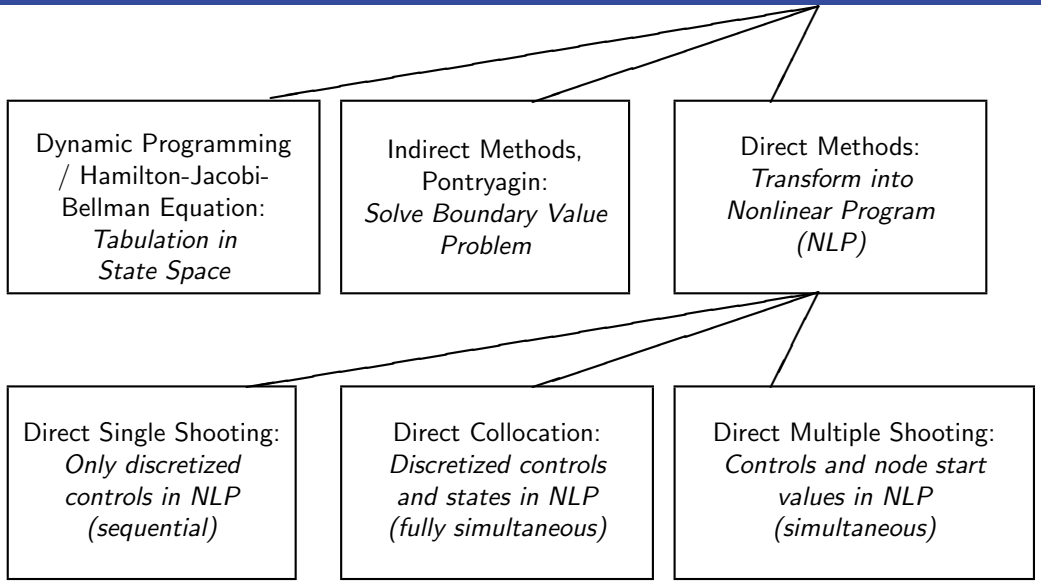
$$J_k(x) = \min_u \max_{w \in \mathbb{W}} \left(L(x,u) + J_{k+1}(f(x,u,w)) \right)$$

starting with terminal cost

$$J_N(x) = E(x)$$

The only additional effort are the evaluations of the worst-cases in each DP step.

Optimal Control Family Tree



Literature

- J.B. Rawlings, D.Q. Mayne & M. Diehl. Model predictive control: theory, computation, and design (2nd Edition). Nob Hill Publishing, 2024
- L.T. Biegler, Nonlinear programming: concepts, algorithms, and applications to chemical processes. SIAM, 2010
- Diehl, M., Ferreau, H. J., & Haverbeke, N. (2009). Efficient numerical methods for nonlinear MPC and moving horizon estimation. In Nonlinear model predictive control: towards new challenging applications (pp. 391-417), Springer, 2009
- M. Diehl, S. Gros, Numerical Optimal Control (draft), 2025
- J. T. Betts: Practical Methods for Optimal Control Using Nonlinear Programming. SIAM, Philadelphia, 2001. ISBN 0-89871-488-5
- A. E. Bryson and Y. C. Ho: Applied Optimal Control, Hemisphere/Wiley, 1975.
- Dimitri P. Bertsekas: Dynamic Programming and Optimal Control. Athena Scientific (2001)
- Dimitri P. Bertsekas: Lessons from AlphaZero for Optimal, Model Predictive, and Adaptive Control. Athena Scientific (2022).
- J. Björnberg and M. Diehl: Approximate robust dynamic programming and robustly stable MPC. Automatica (2006)
- R. Verschueren, et al. "acados—a modular open-source framework for fast embedded optimal control." Mathematical Programming Computation 14.1 (2022): 147-183.