KU LEUVEN

OCP/MPC Workshop 2024
Nonlinear Optimal Control

MISTI MIT Global Experiences

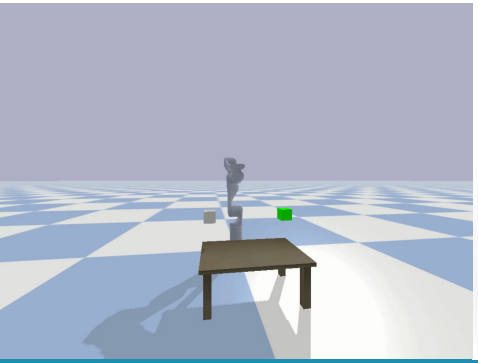
Wilm Decré
wilm.decre@kuleuven.be

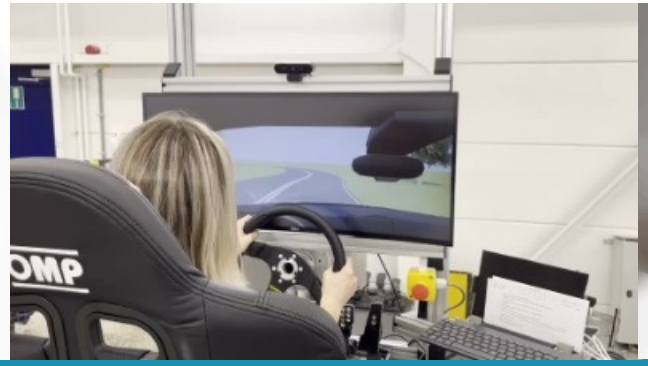
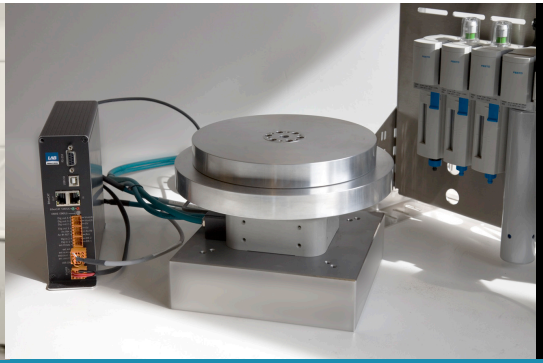
Overview

- **optimal control problems?**
- solution strategies - direct methods
  - shooting methods
  - collocation methods
- special cases and extensions
  - periodic problems
  - free-time problems
  - multi-stage problems

Optimal control

- find the “optimal” trajectory for a dynamical system, given
  - system dynamics, $\dot{x}(t) = f(x(t), u(t), t)$ (~ “state transition function” in the learning community) (here: deterministic)
  - initial state $x(0) = x_0$ (later we will relax this)
  - objective function, e.g., time, control effort, …
  - set of constraints, e.g., no-collision
- open-loop: find optimal control and state trajectory $u^*(t), x^*(t)$ ⟵ topic of today
  - direct methods, indirect methods
- closed-loop: find optimal control policy π^* such that $u^*(x(t), t) = \pi^*(x(t), t)$
  - dynamic programming, learning techniques
- model predictive control
- for notational simplicity, in the remainder of the slides we will omit * and the function arguments if they are clear from the context




Why is optimal control challenging?

- challenges
  1. solve optimal control problems **fast** and **robustly** (global convergence)
     - in a few special cases we can solve optimal control problems directly, but mostly we employ iterative methods
     - high bandwidth/sampling rates, … in an MPC setting
     - expensive nonlinear functions, such as dynamics and no-collision constraints
  2. make optimal control **accessible** for users
- to address these challenges
  - implementations that exploit hardware to accelerate computations

$$\text{CPU time} = \text{instruction count} \times \text{CPI} \times \text{clock cycle time}$$

  - tailored solvers that exploit the structure of optimal control problems
  - accessible software
  - efficient formulations of constrained dynamics

Optimal control problem

optimal control problem with state x, control u, on a horizon from 0 to T:

$$\underset{x(\cdot),u(\cdot)}{\text{minimize}}\; l_T\big(x(T)\big) + l_0\big(x(0)\big) + \int_0^T l(x(t),u(t),t)\mathrm{d}t \quad \longleftarrow \text{stagewise cost}$$

$$\text{subject to}$$

$$\begin{cases} \dot{x} = f(x,u,t) & \longleftarrow \text{system dynamics} \\ g(x,u,t) = 0 & \longleftarrow \text{stagewise equality constraints} \\ h(x,u,t) \leq 0 & \longleftarrow \text{stagewise inequality constraints} \end{cases}$$

with, unless noted otherwise, l, l_T, l_0, f, g and h twice continuously differentiable

this problem is infinite-dimensional… solution methods?

- indirect methods – "*optimize then discretize*"
- direct methods – "*discretize then optimize*"

Overview

- optimal control problems?
- **solution strategies - direct methods**
  - shooting methods
  - collocation methods
- special cases and extensions
  - periodic problems
  - free-time problems
  - multi-stage problems

Direct methods

- *"discretize then optimize"*
  1. discretize the problem using a transcription method
     - shooting
     - collocation
  2. solve the discretized problem

$$\underset{x(\cdot),u(\cdot)}{\text{minimize}}\; l_T\big(x(T)\big) + l_0\big(x(0)\big) + \int_0^T l(x(t),u(t),t)\mathrm{d}t$$
$$\text{subject to}$$
$$\begin{cases} \dot{x} = f(x,u,t) \\ g(x,u,t) = 0 \\ h(x,u,t) \leq 0 \end{cases}$$

Direct methods

- *"discretize then optimize"*
  1. discretize the problem using a transcription method
     - **shooting**
     - collocation
  2. solve the discretized problem

$$\underset{x(\cdot),u(\cdot)}{\text{minimize}}\; l_T\big(x(T)\big) + l_0\big(x(0)\big) + \int_0^T l(x(t),u(t),t)\mathrm{d}t$$
$$\text{subject to}$$
$$\begin{cases} \dot{x} = f(x,u,t) \\ g(x,u,t) = 0 \\ h(x,u,t) \leq 0 \end{cases}$$

Direct shooting

$$\underset{x(\cdot),u(\cdot)}{\text{minimize}}\; l_T\big(x(T)\big) + l_0\big(x(0)\big) + \int_0^T l(x(t),u(t),t)\mathrm{d}t$$

subject to

$$\begin{cases} \dot{x} = f(x,u,t) \\ g(x,u,t) = 0 \\ h(x,u,t) \leq 0 \end{cases}$$

we discretize the controls u on a grid of length K

- here we assume zero-order hold (zoh) controls
- we hence have controls $u[0], u[1], \ldots, u[K-1]$
- we use (numerical) integrators to compute the state and cost (cont. on next slides)

in its basic form, we use the same gridding, and obtain:

$$\underset{x[\cdot],u[\cdot]}{\text{minimize}}\; l_T(x[K]) + l_0(x[0]) + \sum_{k=0}^{K-1} l_d(x[k],u[k],k)$$

subject to

$$\begin{cases} x[k+1] = f_d(x[k],u[k],k) \\ g(x[k],u[k],k) = 0, g_K(x[K]) = 0 \text{ for } k = 0,1,\ldots,K-1 \\ h(x[k],u[k],k) \leq 0, h_K(x[K]) \leq 0 \end{cases}$$

Direct shooting

$$\underset{x[\cdot],u[\cdot]}{\text{minimize}}\; l_T(x[K]) + l_0(x[0]) + \sum_{k=0}^{K-1} l_d(x[k], u[k], k)$$

subject to

$$\begin{cases} x[k+1] = f_d(x[k], u[k], k) \\ g(x[k], u[k], k) = 0, g_K(x[K]) = 0 \text{ for } k = 0, 1, \dots, K-1 \\ h(x[k], u[k], k) \leq 0, h_K(x[K]) \leq 0 \end{cases}$$

- single shooting: we eliminate $x[k]$ for $k = 1, 2, \dots, K$ using the discrete-time state dynamics, and retain only $x[0]$ and the controls as decision variables
- multiple shooting: we keep the states as decision variables
- remark 1: if the initial state is given, it can be eliminated as well
- remark 2: the constraints are only enforced on a grid

Direct shooting

single versus multiple shooting?

single shooting	multiple shooting
smaller problem size	larger problem size, yet specific sparsity pattern that can be exploited
feasible initialization w.r.t. the system dynamics	allows infeasible initialization w.r.t. the system dynamics
	improved distribution of nonlinearities, leading to superior convergence
	some of the underlying computations can be parallelized (more) easily

$\Longrightarrow$in practice: in most cases that we will consider, multiple shooting is preferred

Rockit/CasADi, you can inspect the sparsity with the `spy` function

Direct shooting - integrators

$$\dot{x} = f(x, u, t)$$
$$\text{step size } h = \frac{T}{K}$$

forward (or explicit) Euler

- $x[k+1] = x[k] + hf(x[k], u[k], k) = f_d(x[k], u[k], k)$
- local truncation error (LTE) (assuming exact arithmetic)
  - local, we have: $x[k] = x(kh)$
  - Taylor expansion: $x\big((k+1)h\big) = x(kh) + h\dot{x}(kh) + \frac{1}{2}h^2\ddot{x}(kh) + O(h^3)$
  - LTE: $x\big((k+1)h\big) - x[k+1] = \frac{1}{2}h^2\ddot{x}(kh) + O(h^3)$
  - error is approximately proportional to h^2 (for small h) (assuming a bounded third derivative)
- global truncation error (error over time) is approximately proportional to h (under some mild assumptions)
  - intuition: number of steps for a given integration time is proportional to $\frac{1}{h}$, and error in each step to h^2
  - hence: explicit Euler is said to be “first-order”

explicit Runge-Kutta 4 (RK4)

- $x[k+1] = x[k] + \frac{h}{6}(m_1 + 2m_2 + 2m_3 + m_4)$, with
  - $m_1 = f(x[k], u[k], kh)$
  - $m_2 = f\left(x[k] + \frac{hm_1}{2}, u[k], kh + \frac{h}{2}\right)$
  - $m_3 = f\left(x[k] + \frac{hm_2}{2}, u[k], kh + \frac{h}{2}\right)$
  - $m_4 = f(x[k] + hm_3, u[k], kh + h)$
- a single step is much more expensive to calculate
- but… much more accurate (for small h): global truncation error $O(h^4)$, hence “fourth-order method” (under some mild assumptions)

Direct shooting - integrators

can we do better?

for linear time-invariant systems we can integrate *exactly* (assuming zoh controls) …

- $\dot{x} = Ax + Bu$
- on a single integration interval we have a constant u, hence:
- $\begin{bmatrix} x[k+1] \\ - \end{bmatrix} = e^{\begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix} h} \begin{bmatrix} x[k] \\ u[k] \end{bmatrix} \Rightarrow x[k+1] = A_d x[k] + B_d u[k]$
- routines available in, e.g., `scipy`: `scipy.signal.cont2discrete`

Direct methods

- *"discretize then optimize"*
  1. discretize the problem using a transcription method
     - shooting
     - **collocation**
  2. solve the discretized problem

$$\underset{x(\cdot),u(\cdot)}{\text{minimize}}\; l_T\big(x(T)\big) + l_0\big(x(0)\big) + \int_0^T l(x(t),u(t),t)\mathrm{d}t$$
$$\text{subject to}$$
$$\begin{cases} \dot{x} = f(x,u,t) \\ g(x,u,t) = 0 \\ h(x,u,t) \leq 0 \end{cases}$$

Direct collocation

- again, consider: $\dot{x} = f(x, u, t)$
- define n collocation points $0 \leq c_1 < c_2 < \ldots < c_n \leq 1$ per integration step h
- x_p is a polynomial approximation of x, of degree n, such that
  - $x_p(t_0) = x[k]$
  - $\dot{x}_p(t_i) = f\left(x_p(t_i), u[k], t_i\right)$, with $t_i = t_0 + c_i h$
  - hence: $n + 1$ equations to determine the parameters of x_p
- for the entire horizon we obtain a piecewise polynomial approximation – splines
- basic form: implicit (or backward) Euler, with $n = 1$ and $c_1 = 1$

Direct methods

- *"discretize then optimize"*
  1. discretize the problem using a transcription method
     - shooting
     - collocation
  2. **solve the discretized problem**

$$\underset{x(\cdot),u(\cdot)}{\text{minimize}}\; l_T\big(x(T)\big) + l_0\big(x(0)\big) + \int_0^T l(x(t),u(t),t)\mathrm{d}t$$

$$\text{subject to}$$

$$\begin{cases} \dot{x} = f(x,u,t) \\ g(x,u,t) = 0 \\ h(x,u,t) \leq 0 \end{cases}$$

NLP solvers

- after transcription (shooting or collocation) we obtain a nonlinear programming problem (NLP), as covered in the previous lecture:

$$\begin{aligned} &\underset{x}{\text{minimize}}\ f(x) \\ &\text{subject to} \\ &\begin{cases} g(x) = 0 \\ h(x) \leq 0 \end{cases} \end{aligned}$$

- we can solve this problem using a general-purpose NLP solver, e.g., Ipopt (with a sparse linear solver)
- … or use a specialized solver for the OCP problem class such as Fatrop
  - custom linear solver
  - linear algebra routines performance-optimized for small matrices

$$\begin{array}{cccccccccc} \mathbf{x}_2 & \mathbf{v}_2 & \boldsymbol{\pi}_2 & \mathbf{u}_1 & \mathbf{x}_1 & \boldsymbol{\lambda}_1 & \boldsymbol{\pi}_1 & \mathbf{u}_0 & \mathbf{x}_0 & \boldsymbol{\lambda}_0 & \\ \left[\begin{array}{cccccccccc} \mathbf{Q}_2 & \mathbf{H}_2' & -\boldsymbol{I} & & & & & & & & \mathbf{q}_2 \\ \mathbf{H}_2 & & & & & & & & & & \mathbf{h}_2 \\ -\boldsymbol{I} & & & \mathbf{B}_1 & \mathbf{A}_1 & & & & & & \mathbf{b}_1 \\ & & \mathbf{B}_1' & \mathbf{R}_1 & \mathbf{S}_1' & \mathbf{H}_{1,u}' & & & & & \mathbf{r}_1 \\ & & \mathbf{A}_1' & \mathbf{S}_1 & \mathbf{Q}_1 & \mathbf{H}_{1,x}' & -\boldsymbol{I} & & & & \mathbf{q}_1 \\ & & & \mathbf{H}_{1,u} & \mathbf{H}_{1,x} & & & & & & \mathbf{h}_1 \\ & & & & -\boldsymbol{I} & & & \mathbf{B}_0 & \mathbf{A}_0 & & \mathbf{b}_0 \\ & & & & & & \mathbf{B}_0' & \mathbf{R}_0 & \mathbf{S}_0' & \mathbf{H}_{0,u}' & \mathbf{r}_0 \\ & & & & & & \mathbf{A}_0' & \mathbf{S}_0 & \mathbf{Q}_0 & \mathbf{H}_{0,x}' & \mathbf{q}_0 \\ & & & & & & & \mathbf{H}_{0,u} & \mathbf{H}_{0,x} & & \mathbf{h}_0 \end{array}\right] \end{array}$$

structure of the primal-dual system for $K = 2$ [2]

Remark (to avoid confusion): we overloaded the symbols: x is not the state, but represents all (finite number of) decision variables

Overview

- optimal control problems?
- solution strategies
  - indirect methods
  - direct methods
    - shooting methods
    - collocation methods
- **special cases and extensions**
  - periodic problems
  - free-time problems
  - multi-stage problems

Periodic problems

constraints to impose that the terminal state is equal to the initial state

- trivial if the initial state (and hence also the terminal state) is known
- otherwise it breaks our stagewise structure…

$$\underset{x(\cdot),u(\cdot)}{\text{minimize}}\; l_T\big(x(T)\big) + l_0\big(x(0),u(0)\big) + \int_0^T l(x(t),u(t),t)\mathrm{d}t$$

subject to

$$\begin{cases} \dot{x} = f(x,u,t) \\ g(x,u,t) = 0 \\ h(x,u,t) \leq 0 \\ x(0) = x(T) \end{cases}$$

Periodic problems

how do we deal with this?

- when using a general-purpose NLP solver, the solver can directly deal with this problem
- when using an OCP solver, we can rewrite the problem by augmenting the state
  - $\hat{x} = \begin{bmatrix} x \\ \xi \end{bmatrix}, \hat{f}(\hat{x}, u, t) = \begin{bmatrix} f(x, u, t) \\ 0 \end{bmatrix}, \hat{g}(\hat{x}, u, t) = \begin{bmatrix} g(x, u, t) \\ \xi(0) = x(0) \\ \xi(T) = x(T) \end{bmatrix}$
  - $\hat{h}(\hat{x}, u, t) = h(x, u, t), \hat{l}_T\big(\hat{x}(T)\big) = l_T\big(x(T)\big), \hat{l}_0\big(\hat{x}(0), u(0)\big) = l_0\big(x(0), u(0)\big), \hat{l}(\hat{x}(t), u(t), t) = l(x(t), u(t), t)$
- this results in a standard OCP problem:

$$\underset{\hat{x}(\cdot), u(\cdot)}{\text{minimize}}\ \hat{l}_T\big(\hat{x}(T)\big) + \hat{l}_0\big(\hat{x}(0), u(0)\big) + \int_0^T \hat{l}(\hat{x}(t), u(t), t)\mathrm{d}t$$

$$\text{subject to}$$

$$\begin{cases} \dot{\hat{x}} = \hat{f}(\hat{x}, u, t) \\ \hat{g}(\hat{x}, u, t) = 0 \\ \hat{h}(\hat{x}, u, t) \leq 0 \end{cases}$$

- drawback: larger problem size (the number of state variables doubles!), yet there is structure that can be exploited

Free-time problems

the end time becomes an optimization variable

$$\underset{x(\cdot),u(\cdot),T}{\text{minimize}}\; l_T(x(T),T) + l_0\big(x(0),u(0)\big) + \int_0^T l(x(t),u(t),t)\mathrm{d}t$$

subject to

$$\begin{cases} \dot{x} = f(x,u,t) \\ g(x,u,t) = 0 \\ h(x,u,t) \leq 0 \\ T \geq 0 \end{cases}$$

we don't know the length of the horizon, so how to discretize…?

Free-time problems

how can we deal with this?

we introduce scaled time: $s = \frac{t}{T}$ and then rewrite the problem with s as independent variable, the horizon then runs from $s = 0$ to $s = 1$, and we can transform to an NLP!

$$\underset{x(\cdot),\, u(\cdot),\, T}{\text{minimize}}\; l_T(x(1), T) + l_0\big(x(0), u(0)\big) + \int_0^1 l(x(s), u(s), sT)\,\mathrm{d}s$$

subject to

$$\begin{cases} \dfrac{dx}{ds} = T f(x, u, sT) \\ g(x, u, sT) = 0 \\ h(x, u, sT) \leq 0 \\ T \geq 0 \end{cases}$$

a general NLP solver can directly solve this problem
for an OCP solver, we introduce an additional state variable
variants are possible with non-uniform grids

Multi-stage problems

- multiple optimal control problems “stitched together”
  - e.g., discontinuous change in dynamics or other constraints

$$\underset{x(\cdot),u(\cdot),T_1,T_2}{\text{minimize}} \; l_{T_2}\big(x(T_2)\big) + \int_{T_1}^{T_2} l_2(x(t),u(t),t)\mathrm{d}t + l_{T_1}\big(x(T_1)\big) + l_0\big(x(0)\big) + \int_0^{T_1} l_1(x(t),u(t),t)\mathrm{d}t$$

subject to

$$\begin{cases} \dot{x} = f_1(x,u,t) \\ g_1(x,u,t) = 0 \\ h_1(x,u,t) \leq 0 \\ T_1 \geq 0 \\ \dot{x} = f_2(x,u,t) \\ g_2(x,u,t) = 0 \\ h_2(x,u,t) \leq 0 \\ T_2 \geq T_1 \end{cases}$$

imposed during first stage from 0 to T_1

imposed during second stage from T_1 to T_2

References and further reading

- [1] Rawlings, J. B., Mayne, D. Q., & Diehl, M. M. (2017). *Model predictive control: theory, computation, and design* (2nd ed.). Nob Hill Publishing.
- [2] Vanroye, L., Sathya, A., De Schutter, J., & Decré, W. (2023). FATROP : A Fast Constrained Optimal Control Problem Solver for Robot Trajectory Optimization and Control. *ArXiv.Org*. https://doi.org/10.48550/arxiv.2303.16746