Optimal Vehicle Maneuvers — Lecture 2

Lars Nielsen[1] & Björn Olofsson[2]

[1]Division of Vehicular Systems, Linköping University, Sweden
[2]Department of Automatic Control, Lund University, Sweden

Lecture 2

Outline

Course structure

Lectures in Block 1:

- Introduction (Lecture 1)
  - Autonomous driving at large
  - Course outline
  - Introduction to Optimal Vehicle Motion Control
- Optimization framework (Lecture 2)
  - Solving the Optimal Control Problem
  - Introduction, numerical tools, modeling issues
- Computing optimal maneuvers (Lectures 3-4)
  - Maneuvers: hairpin, curve, moose test (ISO-3888-2), ...
  - Vehicle models (particle, ST, DT)
  - Minimum time
  - Entry speed, v_0, and exit speed v_f
  - Recovery
  - Racing
  - Narrow lane and path tolerance

Recommended Reading for This Lecture

- Limebeer, D. J., & A. V. Rao: "Faster, higher, and greener: Vehicular optimal control". IEEE Control Systems Magazine, 35(2), 36–56, 2015.
- For a more mathematical treatment of the topic of numerical optimal control and further reading on the methods presented in this lecture: Chapter 8 in Rawlings, J. B., D. Q. Mayne, & M. Diehl: *Model Predictive Control: Theory, Computation, and Design*. 2nd Edition. Nob Hill Publishing, 2017.

Recapitulation: Optimization Formulation from Lecture 1

- The optimal control problem for minimization of a criterion J for a maneuver is formulated as:

$$\begin{aligned} \text{minimize} \quad & J \\ \text{subject to} \quad & A(u, \dot{u}, ...) \leq 0 \\ & F_c x(0) = \tilde{x}_0, \; G_c x(t_f) = \tilde{x}_f, \\ & f(X_p, Y_p) \leq 0 \\ & \dot{x} = G(x, y, u), \; h(x, y, u) = 0 \end{aligned}$$

- Optimization criterion and situation awareness: J and f.
- Actuator constraints: $A(u, \dot{u}, ...)$ are constraints on, e.g., steering angle δ (or $\dot{\delta}$), and wheel torques T_i and $\dot{T}_i$.
- Vehicle model: Chassis dynamics, wheel dynamics, and tire model in $\dot{x} = G(x, y, u)$ and $h(x, y, u) = 0$.

Examples of Optimal Control Problems

Examples:

- *Motion planning for autonomous vehicles and robots*,
- Optimal battery-charging planning for autonomous electric vehicles,
- Efficient electric motor and engine control,
- Traffic-flow optimization in city environments.

Involves dynamics (differential equations) that need to be considered. Dynamic optimization.

Introduction

- Numerical methods for dynamic optimization, since many real-world problems are intractable with analytical methods.
- Developments in methods and tools the last decade, enabling solution times decreasing from hours to seconds also for large-scale problems.
- Optimization algorithms often find loopholes in model, finding the correct combination of models and formulations is essential.
- Many optimization algorithms put requirements on modeling and formulation (e.g., differentiation).
- Often involving continuous-time dynamics (infinite number of decision variables).

Model Description

- Explicit ordinary differential equation (ODE) system:

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

- The variables are defined as: t – time, x – states, u – inputs.
- Independent variable considered in this lecture is time t, but also other variables like distance or a path coordinate possible.
- Also possible with differential-algebraic equation systems in the formulation (e.g., Modelica models), but not considered further in this lecture.

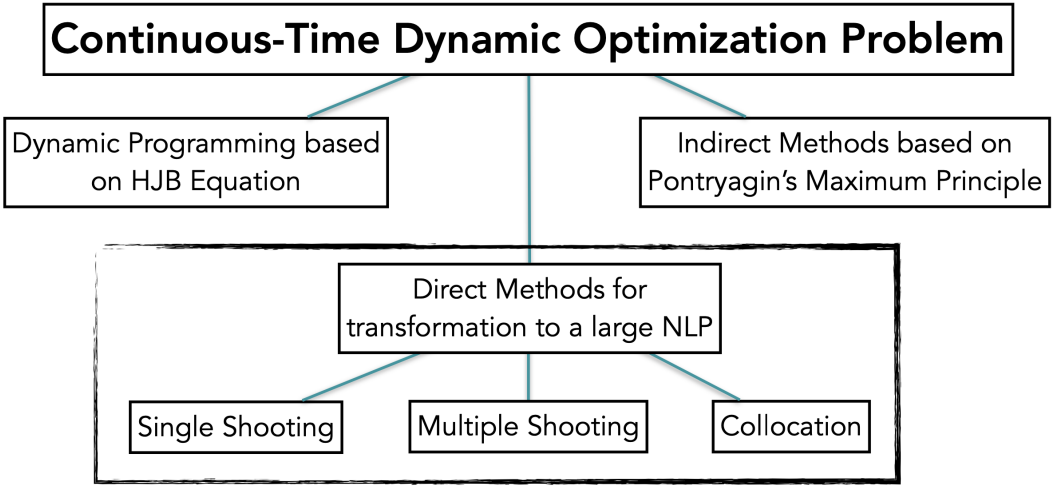
The Optimization Problem

- Introduce the Lagrange integrand L and the Mayer term Γ in the objective function of the optimization problem.
- The sets $\mathbb{X}$, $\mathbb{U}$, and $\mathbb{X}_T$ define the constraints on the states and inputs.
- An optimization problem over the time horizon $[0, T]$ based on ODE dynamics can be formulated as:

$$\begin{aligned} \text{minimize} \quad & \int_0^T L(x(t), u(t))\, \mathrm{d}t + \Gamma(x(T)) \\ \text{subject to} \quad & x(0) = x_0,\ \dot{x}(t) = f(t, x(t), u(t)), \\ & x(t) \in \mathbb{X},\ u(t) \in \mathbb{U},\ x(T) \in \mathbb{X}_T,\ t \in [0, T] \end{aligned}$$

- Terminal time T possibly unknown and thus a decision variable in the optimization, and sometimes also unknown parameters to be found.

Overview of Solution Strategies[1]



[1] Adapted from: Diehl, M: Numerical Optimal Control, Optec, K.U. Leuven, Belgium, 2011.

Solution Methods for Continuous-Time Optimal Control Problems

- Two major approaches to discretization related to continuous-time optimization problems:
  - Discretize the control inputs and reformulate optimization problem in initial state and control inputs (sequential).
  - Discretize both control inputs and states and keep all variables in the optimization (simultaneous).

Direct Methods—The Overall Idea

- Continuous-time optimal control with infinitely many decision variables:

$$\begin{aligned} \text{minimize} \quad & \int_0^T L(x(t), u(t))\,\mathrm{d}t + \Gamma(x(T)) \\ \text{subject to} \quad & x(0) = x_0,\ \dot{x}(t) = f(t, x(t), u(t)), \\ & x(t) \in \mathbb{X},\ u(t) \in \mathbb{U},\ x(T) \in \mathbb{X}_T,\ t \in [0, T] \end{aligned}$$

- A discretization process leads to an optimization problem with a finite (though possibly large) set of variables, a non-linear programming problem:

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

Considerations in the Formulation

- Extending state dimension allows penalty in objective function and constraints on derivative of inputs.
  - Example: If u is the actual input to the system, introduce $\dot{u}$ as a virtual input in the optimization problem formulation.
- Cost function containing an integral (Lagrange integrand) could be expressed as the value of a state at the terminal time T.
  - Enables use of same discretization for objective as state dynamics.
  - Example: With the desired objective $\int_0^T u(t)^2 \, \mathrm{d}t$, a corresponding state formulation would be $c(T)$, where $\dot{c} = u^2$.

Direct Methods—General Procedure

- First discretize the optimization problem, then optimize to find an approximate solution to the original continuous-time optimization problem.
- Transform the infinite-dimensional optimization problem to a finite-dimensional non-linear program (NLP) by discretization of the control inputs and possibly states.
- Then solve the (typically large) NLP, utilizing sparsity.
- Focus on direct simultaneous methods in this lecture (well-proven track record in many applications).

Explicit vs. Implicit Integration

- Consider an explicit ODE:

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

- Numerical integration with explicit or implicit Runge-Kutta methods (with step size h and parameters a, b, c).
- Implicit methods imply (nonlinear) equation solving.
- The different methods imply different stability and accuracy properties.

Explicit integration method:

$$\begin{aligned}
x_{n+1} &= x_n + h\sum_{i=1}^{s} b_i k_i, \\
k_1 &= f(t_n, x_n), \\
k_2 &= f(t_n + c_2 h, x_n + h(a_{2,1}k_1)), \\
&\vdots \\
k_s &= f(t_n + c_s h, x_n + h(a_{s,1}k_1 + \ldots + a_{s,s-1}k_{s-1}))
\end{aligned}$$

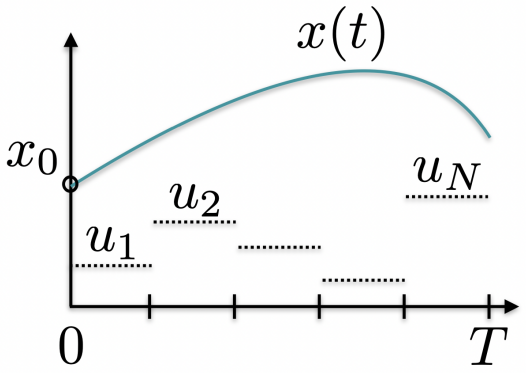
Implicit integration method:

$$\begin{aligned}
x_{n+1} &= x_n + h\sum_{i=1}^{s} b_i k_i, \\
k_1 &= f(t_n + c_1 h, x_n + h(a_{1,1}k_1 + \ldots + a_{1,s}k_s)), \\
k_2 &= f(t_n + c_2 h, x_n + h(a_{2,1}k_1 + \ldots + a_{2,s}k_s)), \\
&\vdots \\
k_s &= f(t_n + c_s h, x_n + h(a_{s,1}k_1 + \ldots + a_{s,s}k_s))
\end{aligned}$$

Single Shooting (1/2)

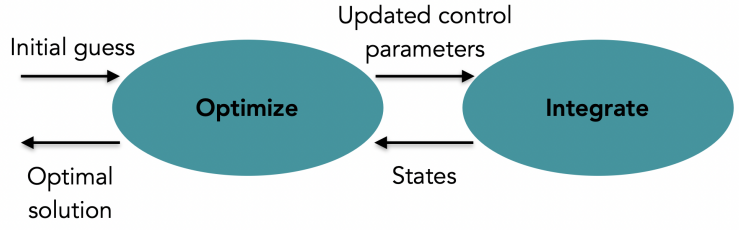
Basic idea of single shooting:

- Discretize control inputs (piecewise constant in the figure to the right).
- Start with an initial guess of control inputs and integrate dynamics forward in time with these inputs.
- Iteratively update control inputs and re-simulate dynamics forward.



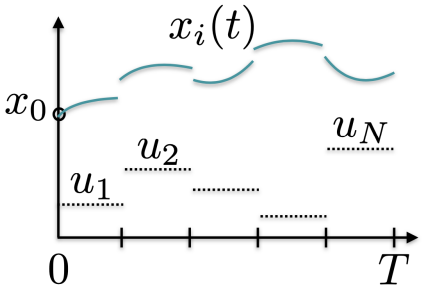
Single Shooting (2/2)

- Sequential approach that optimizes over the control parameters (typically assuming piecewise constant control).
- One of the most straightforward methods to implement, though challenging with unstable system dynamics and handling of state constraints.



Multiple Shooting

- Divide the time horizon into elements and make a piecewise approximation of the input.
- Add continuity constraints (equality) at the element junctions for the states.
- A hybrid between sequential and simultaneous approach.
- Explicit Runge-Kutta methods common for the integration of the states in each element.
- The division into elements implies better numerical stability (in particular for unstable system dynamics).
- Allows initialization of state trajectories and handling of path constraints.



Collocation

- As in multiple shooting, divide the time horizon into elements.
- Discretize both state and input trajectories and include numerical integration conditions as equality constraints in the optimization (collocation equations), compare with implicit integration methods.
- Define a number of collocation points within each element.
- Different schemes common in literature: Gauss-Legendre, Radau, and Lobatto.
- Pseudo-spectral methods: only one element over the entire time horizon $[0, T]$, but high-order polynomials for the interpolation polynomials (i.e., many collocations points).

Collocation — Example (1/2)

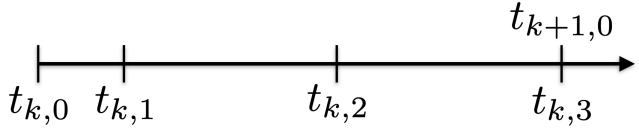
- Approximate the state trajectories with piecewise third-order polynomials and collocation points using a Radau scheme.
- Points distributed in the normalized interval $[0, 1]$:

$$\tau = \begin{pmatrix} 0 & 0.1551 & 0.6449 & 1 \end{pmatrix}$$

- Introduce a uniform width of each element

$$t_k = kh, \; k = 0, \ldots, N$$

- Collocation points illustrated graphically (non-uniform!)



Collocation — Example (2/2)

- Within each element, a Lagrange polynomials basis is used to interpolate the values at the collocation points

$$L_i(\tau) = \prod_{j=0, j\neq i}^{3} \frac{\tau - \tau_j}{\tau_i - \tau_j}$$

- The state trajectory is then approximated as $(t_k = kh,\ k = 0, \ldots, N)$

$$x_\ell(t) = \sum_{i=0}^{3} L_i\left(\frac{t - t_k}{h}\right) x_{k,i}, \quad t \in [t_k, t_{k+1}]$$

- Differentiation with respect to time gives

$$\dot{x}_\ell(t_{k,j}) = \frac{1}{h} \sum_{i=0}^{3} \underbrace{\dot{L}_i(\tau_j)}_{C_{i,j}} x_{k,i}$$

An NLP Problem

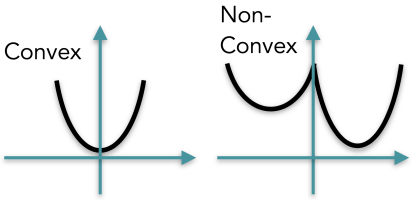
- The direct methods for discretization result in a (typically large) non-linear programming (NLP) problem on the format:

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

- This NLP problem can be solved using various methods, e.g., interior point (IP) and sequential quadratic programming (SQP).
- Barrier-based methods popular for non-linear programming problems in the 1960s, then SQP-like methods were dominant until the middle of the 1980s when new results on IP methods were published and spurred a renewed interest in IP methods for NLPs.

Convex vs. Non-Convex Optimization

- How difficult is it to solve an NLP?
- Difference in optimization is rather between convex and non-convex, not between linear and non-linear.
- Local and global optima of the optimization problem.
- Convex optimization problem (convex feasible set and convex objective function): a local minimum is also a global minimum.



Computing Derivatives—Automatic Differentiation

- Derivatives often used in solution of optimization problem (recall from previous courses in calculus that local optima of a function can be found using the derivative).
- Automatic differentiation (AD) structured way of computing derivatives with machine precision.
- Chain rule for differentiation used to decompose the problem into smaller elementary operations, avoiding establishing explicit symbolic expressions.
- Provides gradients (first-order derivatives) and Hessians (second-order derivatives) for solution of the NLP problem using Newton-based methods.
- Forward or backward mode can give very different performance.
  - Forward mode when #inputs $\ll$ #outputs.
  - Backward mode when #inputs $\gg$ #outputs.
- Example: Backpropagation in training of neural networks.

Automatic Differentiation—An Example

- Compute gradient of F using AD in forward mode:

$$F(x_1, x_2) = x_1 x_2 + \cos(x_1)$$

- Decompose into elementary operations:

$$\begin{aligned} x_3 &= x_1 x_2, & \dot{x}_3 &= \dot{x}_1 x_2 + x_1 \dot{x}_2, \\ x_4 &= \cos(x_1), & \dot{x}_4 &= -\sin(x_1)\dot{x}_1, \\ x_5 &= x_3 + x_4 & \dot{x}_5 &= \dot{x}_3 + \dot{x}_4 \end{aligned}$$

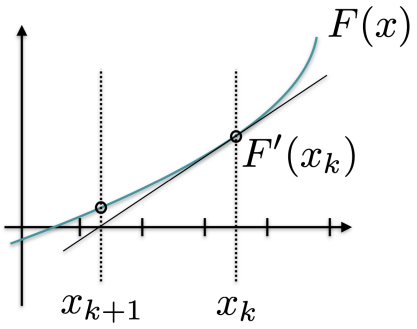
- The final row in the right column is the desired derivative. Performing these computations twice for the so called seeds

$$\dot{x}_1 = 1,\ \dot{x}_2 = 0 \text{ resp. } \dot{x}_1 = 0,\ \dot{x}_2 = 1$$

  gives the desired gradient.
- Only one sweep using backward mode AD.

Newton's Method (1/2)

- Iterative method for finding the roots of a function F, i.e., an x such that $F(x) = 0$, based on a starting point x_0.
- Iteratively linearize the function around the current value x_k and subsequently update value to x_{k+1}.



Newton's Method (2/2)

- Take a step in the computed direction (here assuming a square Jacobian matrix):

$$F(x_k) + J(x_k)(x - x_k) = 0, \text{ where } J(x_k) = \frac{\partial F}{\partial x}(x_k) \Rightarrow$$
$$x_{k+1} = x_k - J(x_k)^{-1} F(x_k)$$

- New step computed by solving the linear equation system:

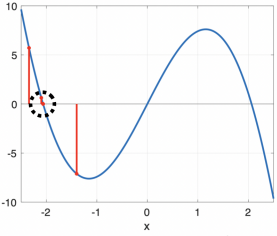
$$J(x_k)\Delta x = -F(x_k), \text{ where } \Delta x = x_{k+1} - x_k$$

- Can be used for finding local optima to optimization problems—initialization strategies for variables important.

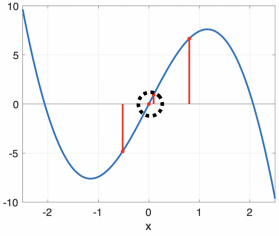
Newton's Method—Example

- An example for finding the roots $F(x) = 0$. Note the dependence on the starting value of x.

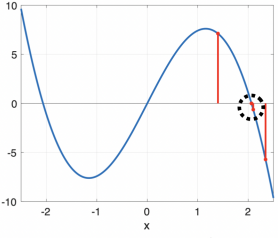
$$F(x) = 10\sin(x) - x^3, \quad F'(x) = 10\cos(x) - 3x^2$$



$x_0 = -1.4$



$x_0 = 0.8$



$x_0 = 1.4$

Initialization and Initial Guess for Optimization Variables

- Which local minimum does the algorithm converge to? How good is that minimum?
- Efficient methods exist for finding a *global minimum* to convex optimization problems and for finding *local minima* to non-convex optimization problems.
- Initialization of optimization variables important for non-convex optimization.
- Simulation with some nominal inputs common approach (e.g., driver models for vehicles).
- One approach to investigate number of local minima is variation of the initial guess for the optimization variables.
- Important with settings of tolerances in the NLP problem solver.

KKT Conditions

- NLP problem to be solved

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

- Introduce the Lagrangian $\mathcal{L}$ according to

$$\mathcal{L}(x, \lambda, \nu) = f(x) + \lambda^{\mathrm{T}} g(x) + \nu^{\mathrm{T}} h(x)$$

- First-order optimality conditions, given certain technical conditions on the constraints (constraint qualification), by Karush, Kuhn, and Tucker (KKT):

$$\begin{aligned} \nabla_x \mathcal{L}(x^*, \lambda^*, \nu^*) &= 0, \ \text{(stationarity)} \\ g(x^*) &= 0, \ \text{(primal feasibility)} \\ h(x^*) &\leq 0, \ \text{(primal feasibility)} \\ \nu^* &\geq 0, \ \text{(complementarity conditions)} \\ \nu_i^* h_i(x^*) &= 0, \ i = 1, \ldots, n_h, \ \text{(complementarity conditions)} \end{aligned}$$

Finding a Solution to the KKT Conditions

- For equality-constrained NLP problems (i.e., $h(x) = 0$), the KKT conditions give a nonlinear system of equations to be solved:

$$\begin{pmatrix} \nabla_x \mathcal{L}(x, \lambda) \\ g(x) \end{pmatrix} = 0$$

- Introduce the variables and function:

$$\tilde{x} = \begin{pmatrix} x \\ \lambda \end{pmatrix}, \quad F(\tilde{x}) = \begin{pmatrix} \nabla_x \mathcal{L}(x, \lambda) \\ g(x) \end{pmatrix}$$

- Application of Newton's method for this case gives the iterations as the solution of the linear equation system:

$$\begin{pmatrix} \nabla_x \mathcal{L}(x_k, \lambda_k) \\ g(x_k) \end{pmatrix} + \begin{pmatrix} \nabla_x^2 \mathcal{L}(x_k, \lambda_k) & \nabla g(x_k) \\ \nabla g(x_k)^{\mathrm{T}} & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta \lambda \end{pmatrix} = 0$$

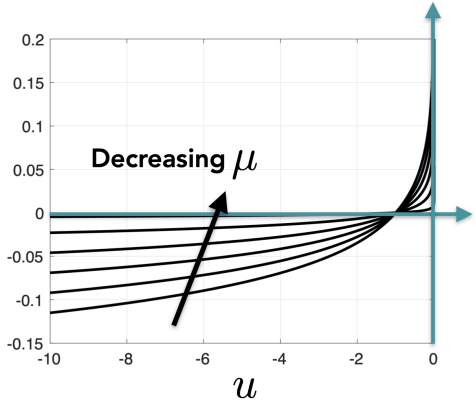
- Partial Newton step with line-search or trust-region strategies to ensure intended decrease of a specified measure.
- Quasi-Newton methods with approximate Hessian exist (of which BFGS is one of the most common).

Interior-Point Methods—Barrier Functions

- How to handle the inequality constraints?
- Recall the constraints $h(x) \leq 0$ in the NLP problem formulation.
- Consider the following approximation of a barrier function with input u:

$$-\mu \log(-u)$$

- The approximation of the barrier improves for decreasing values of the parameter μ.



Interior-Point Methods—Primal-Dual Formulation

- Interior-point methods with barrier functions for inequalities—move inequality constraints to objective function with barrier function and positive parameter μ:

$$\begin{aligned} \text{minimize} \quad & f(x) - \mu \sum_{i=1}^{n_h} \log(-h_i(x)) \\ \text{subject to} \quad & g(x) = 0 \end{aligned}$$

- Could be solved as an equality-constrained problem, but often Lagrange variables for inequalities kept for numerical stability.
- Log-barrier approach corresponds to a smooth approximation of the KKT system:

$$\begin{aligned} \nabla_x \mathcal{L}(x^*, \lambda^*, \nu^*) &= 0, \\ g(x^*) &= 0, \\ \nu_i^* h_i(x^*) &= -\mu, \ i = 1, \ldots, n_h \end{aligned}$$

- Primal-dual variant of IP method solves the above KKT system for decreasing values of barrier parameter μ, while ensuring that the inequalities $\nu > 0, \ h(x) < 0$ hold during the iterations.
- IPOPT is a state-of-the-art implementation of such kind of NLP solver.

Sequential Quadratic Programming

- Alternative to IP methods: Sequential quadratic programming (SQP).
- Iteratively applies a linearization to the equality constraints and the inequality constraints around the current solution (x_k, λ_k, ν_k) to obtain the quadratic program (QP):

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}(x - x_k)^{\mathrm{T}} \nabla_x^2 \mathcal{L}(x_k, \lambda_k, \nu_k)(x - x_k) + \nabla f(x_k)^{\mathrm{T}}(x - x_k) \\ \text{subject to} \quad & g(x_k) + \nabla g(x_k)^{\mathrm{T}}(x - x_k) = 0, \\ & h(x_k) + \nabla h(x_k)^{\mathrm{T}}(x - x_k) \leq 0 \end{aligned}$$

- If no inequality constraints ($h(x) = 0$), equivalent to Newton's method applied to the KKT conditions.
- The QP can be solved using IP methods or active-set methods.

Initialization/Warm Start in IP Methods vs. SQP Methods (1/2)

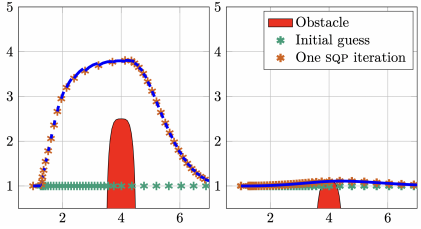
- IP methods have turned out to perform well on large-scale optimization problems, since structure and sparsity in the linear equation systems could be exploited.
- Initialization or warm start of optimization problem variables using a nearby solution is of interest.
- The optimization algorithm has to identify which inequality constraints that are active in the solution, in fact one of the main challenges.

Initialization/Warm Start in IP Methods vs. SQP Methods (2/2)

- IP methods in general more difficult to warm start, since new optimization problem solved with different μ at each new iteration.
- IP methods follow the so called barrier trajectory to the final solution, comparably insensitive to initial values of optimization variables.
- SQP methods have higher potential for warm starting since active inequality constraints are attempted to be estimated explicitly (an active-set type of method).
- SQP methods of interest in MPC where similar problems are solved sequentially, and also typically comprising a lower number of optimization variables suitable for this kind of methods.

Homotopy Methods[2] and SQP Methods

- Gradually introducing constraints using a homotopy parameter (compare with IP methods with logarithmic barrier interpretation).
  - Example: Constraints in a motion-planning problem. Start with a simple problem without constraints, and then gradually introduce the obstacle using the homotopy parameter to finally reach the desired configuration.
- Application of SQP method allows warm starting for next iteration with previous solution.
- Example from PhD Thesis by Kristoffer Bergman:



[2]Bergman, K: "Exploiting Direct Optimal Control for Motion Planning in Unstructured Environments", PhD Thesis, Div. Automatic Control, Linköping Univ., 2021.

Evaluation of Solution

- Evaluation of quality of the computed solution is an important aspect.
- Numerical accuracy—How well do the solution trajectories satisfy the dynamic state equations?
- For offline applications, simulation of the system dynamics with the resulting optimal inputs and evaluate how well the trajectories match is a possibility.
- MPC applications often comprise a trade-off between solution quality and computational time.

Tools for Numerical Optimization

- In this course we will focus on two different tools for solving dynamic optimization problems numerically:
  - CasADi (`https://web.casadi.org`).
  - YOP (`https://www.yoptimization.com`, developed by Viktor Leek in his PhD Thesis at Div. Vehicular Systems), dedicated introduction/exercise in the next block.

CasADi

- Open-source tool for numerical optimization and optimal control.
- Includes efficient algorithms for automatic differentiation and computing Jacobians and Hessians in forward and backward mode.
- Interfaces to various NLP solvers (including IPOPT) and numerical integrators for ODEs/DAEs.

Other Tools

- Examples of other tools for optimization:
  - ACADO,
  - CVX,
  - YALMIP, and
  - JModelica.org.

Conclusions

- Numerical optimization has become a useful tool for offline or online finding optimal control inputs and designing high-performance controllers in many areas.
- Good computer tools and software are available for solving these kinds of problems, but insights and basic knowledge about methods used in the tools are beneficial when struggling with convergence or interpreting results.

Next Lecture

Next lecture: Lars will talk about other optimization criteria, maneuvers, and models.

References (1/2)

- Andersson, J., J. Gillis, G. Horn, and J. B. Rawlings, & M. Diehl: "CasADi–A software framework for nonlinear optimization and optimal control", Mathematical Programming Computation, 2018.
- Bergman, K: "Exploiting Direct Optimal Control for Motion Planning in Unstructured Environments", PhD Thesis, Div. Automatic Control, Linköping Univ., 2021.
- Diehl, M: Numerical Optimal Control, Optec, K.U. Leuven, Belgium, 2011.
- Diehl, M., Bock, H. G., & Schlöder, J. P: "A real-time iteration scheme for nonlinear optimization in optimal feedback control". SIAM Journal on Control and Optimization, 43(5), 1714–1736, 2005.
- Forsgren, A.: "On warm starts for interior methods", In: IFIP Conference on System Modeling and Optimization, Springer, Boston, MA, 51–66, 2005.
- Forsgren, A., Gill, P. E., & Wright, M. H.: "Interior methods for nonlinear optimization", SIAM review, 44(4), 525–597, 2002.

References (2/2)

- Limebeer, D. J., & A. V. Rao: "Faster, higher, and greener: Vehicular optimal control". IEEE Control Systems Magazine, 35(2), 36–56, 2015.
- Nocedal, J., & S. Wright: Numerical Optimization. Springer, 2006.
- Rawlings, J. B., D. Q. Mayne, & M. Diehl: Model Predictive Control: Theory, Computation, and Design. 2nd Edition. Nob Hill Publishing, 2017.
- Wächter, A. & L. T. Biegler: "On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming", Mathematical Programming, 106(1):22–57, 2006.