Dynamic Optimization as a Tool for Motion Planning and Control

TSFS12: Autonomous Vehicles – Planning, Control, and Learning Systems

Lecture 5: Björn Olofsson <bjorn.olofsson@liu.se>

LINKÖPING UNIVERSITY

Purpose of this Lecture

- Give a background on **dynamic optimization**, particularly with respect to **applications** in motion planning and control and the **main ideas of the methods** used:
  - motivation and challenges,
  - fundamental concepts and methods,
  - application examples.
- Demonstrate a **case-study** for optimization of **motion primitives**.
  - Connected to Hand-in Exercise 2.

Expected Take-Aways from this Lecture

- Be familiar with **basic concepts in optimization** and **common methods for numerical optimal control** for systems described by continuous-time dynamic models.
  - Not expected to get insights into all details of the treated methods, primarily on a **general level** for **understanding of their applicability**.
- Have knowledge about **different applications of dynamic optimization** for **motion planning and control** for **autonomous vehicles**.

Literature Reading

The following book and article sections are the main reading material for this lecture. References to further reading are provided throughout the slides and at the end of the lecture slides.

- 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.

Outline of the Lecture

- **Introduction** to **dynamic optimization** and **application examples**.
- **Methods** and **concepts** for **solving continuous-time optimal control problems** using **numerical methods.**
- **Case study**: computation of optimal motion primitives.

Context in the Architecture for an Autonomous Vehicle

Dynamic optimization is possible to apply at all layers in the architecture for different tasks (with varying constraints on real-time computational cost).

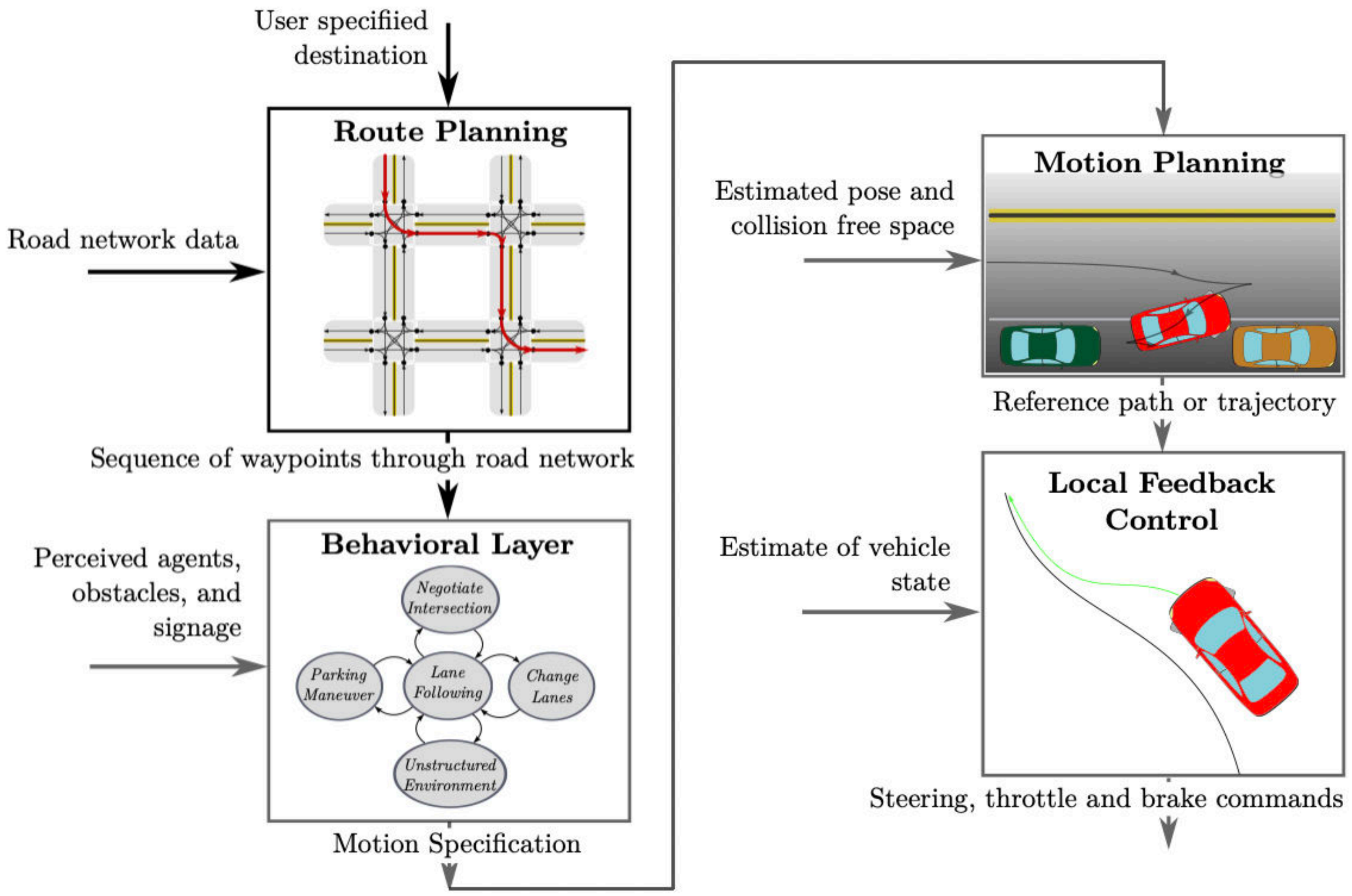
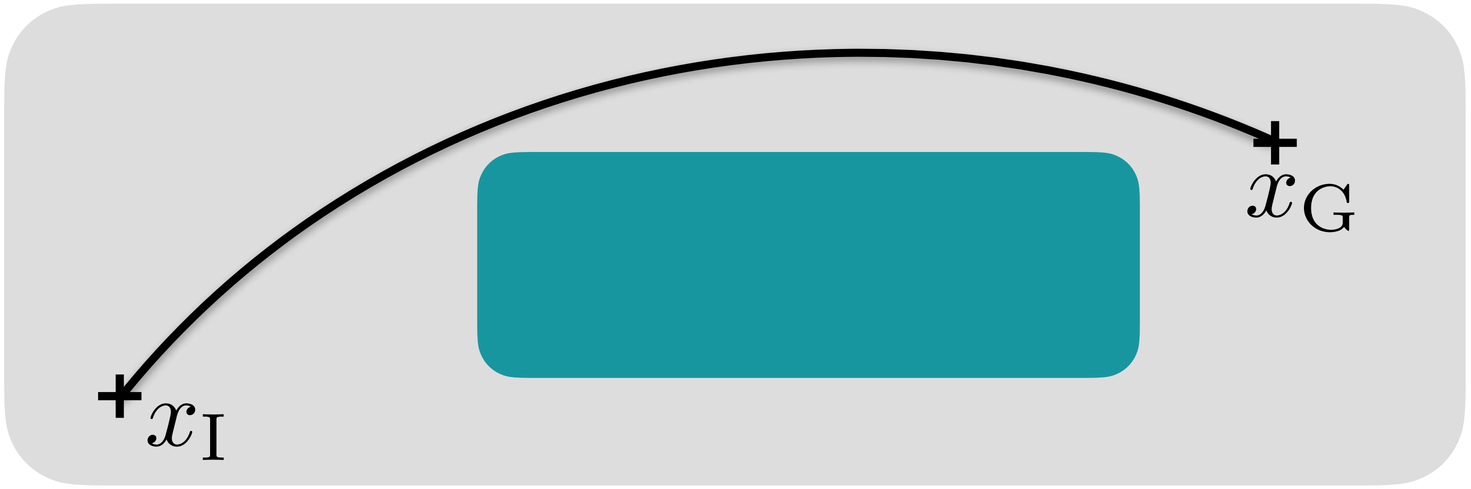

User specified destination
Route Planning
Road network data
Sequence of waypoints through road network
Behavioral Layer
Perceived agents, obstacles, and signage
Negotiate Intersection
Parking Maneuver
Lane Following
Change Lanes
Unstructured Environment
Motion Specification
Motion Planning
Estimated pose and collision free space
Reference path or trajectory
Local Feedback Control
Estimate of vehicle state
Steering, throttle and brake commands

Figure from: Paden, B., Čáp, M., Yong, S. Z., Yershov, D., & Frazzoli, E: "A survey of motion planning and control techniques for self-driving urban vehicles". IEEE Transactions on Intelligent Vehicles, 1(1), 33-55, 2016.

Introduction to Dynamic Optimization and Application Examples

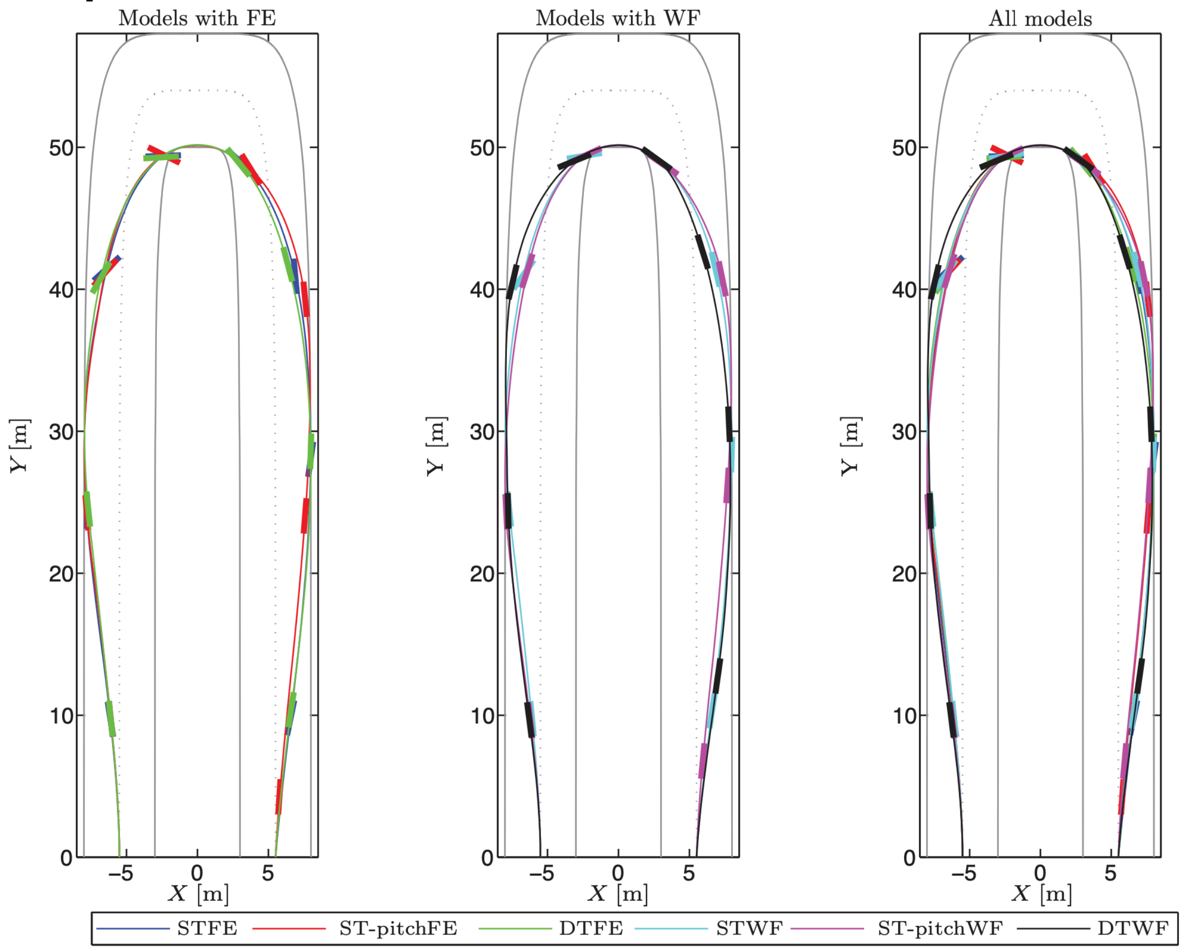
Recall Motion-Planning Problem from Lecture 4

- Compute a strategy for transferring a vehicle **from an initial state to another desired state**.
- Constraints on **control inputs and states**, as well as a **performance criterion** to be fulfilled.



Dynamic Optimization – Examples (1/3)

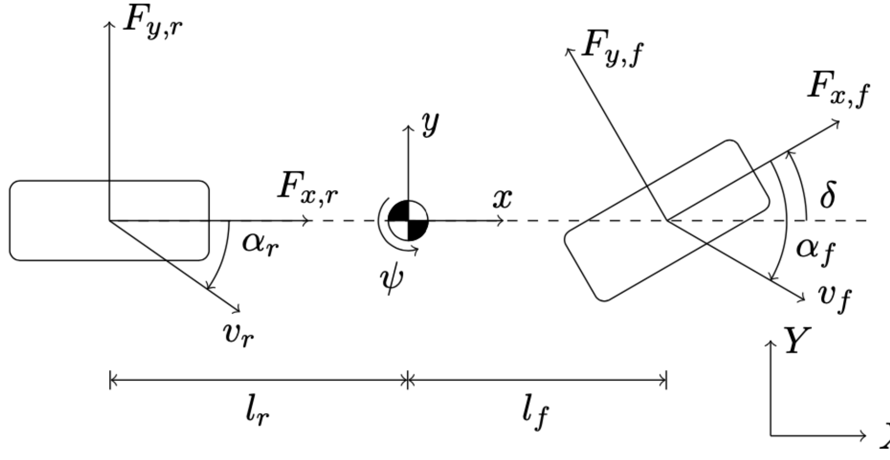
- Vehicle maneuvers at-the-limit of tire friction for development of **future safety systems** for **autonomous vehicles**.
- Traverse the **hairpin turn** in as short time as possible.
- The plot shows results for **different vehicle models.**



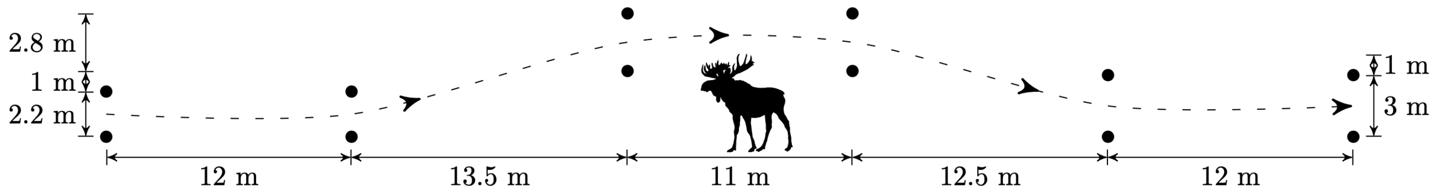
K. Berntorp, B. Olofsson, K. Lundahl, & L. Nielsen: "Models and methodology for optimal trajectory generation in safety-critical road-vehicle manoeuvres", Vehicle System Dynamics, 52(10):1304–1332, 2014.

Dynamic Optimization – Examples (2/3)

- Perform an **avoidance maneuver**, while spending as short time as possible in the opposite driving lane.
- **Single-track vehicle model** combined with **tire–road friction model**.



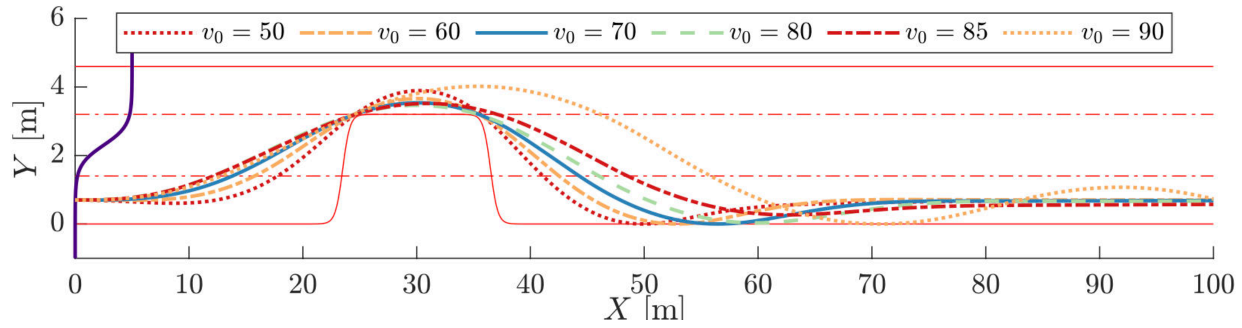
$$m(\dot{v}_x - v_y\dot{\psi}) = F_{x,f}\cos(\delta) + F_{x,r} - F_{y,f}\sin(\delta)$$
$$m(\dot{v}_y + v_x\dot{\psi}) = F_{y,f}\cos(\delta) + F_{y,r} + F_{x,f}\sin(\delta)$$
$$I_Z\ddot{\psi} = l_f F_{y,f}\cos(\delta) - l_r F_{y,r} + l_f F_{x,f}\sin(\delta)$$

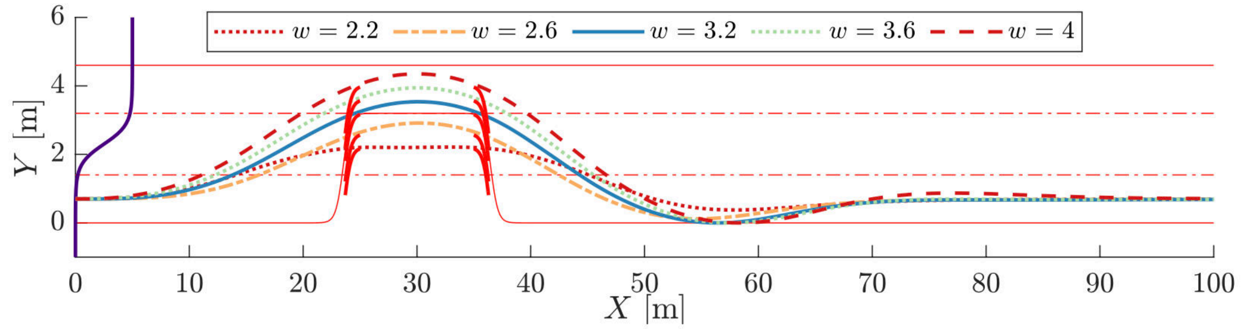


P. Anistratov, B. Olofsson, & L. Nielsen: "Lane-deviation penalty formulation and analysis for autonomous vehicle avoidance maneuvers", Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering, 235(12), 2021.

Dynamic Optimization – Examples (3/3)

- Perform an **avoidance maneuver**, while spending as short time as possible in the opposite driving lane.
- **Vary the initial velocity** and the **geometry of the obstacle** and study the resulting **optimal vehicle maneuvers**.





P. Anistratov, B. Olofsson, & L. Nielsen: "Lane-deviation penalty formulation and analysis for autonomous vehicle avoidance maneuvers", Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering, 235(12), 2021.

Dynamic Optimization – Background (1/2)

- Finding solutions to optimization problems involving **system dynamics** (e.g., the motion equations of an autonomous vehicle).
- Optimal control for finding input signals and state trajectories for optimizing a **performance criterion**, while fulfilling model dynamics and often also other requirements.
- Sometimes also **unknown parameters** to be found.
- The methods in this lecture are important for **model predictive control** (MPC) (see Lecture 7).
- Moving-horizon estimation (MHE) (dual problem to MPC).

Dynamic Optimization – Background (2/2)

- In this lecture the focus is on **numerical methods** for dynamic optimization, since many real-world problems are intractable with analytical methods.
- **Independent variable** considered in this lecture is time t, but also other variables like distance possible.
- 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.

System Dynamics (1/2)

- The **motion equations** to be considered are described by **differential equations** for the states in continuous time (or difference equations in discrete time), see Lecture 3.
- Possibly also **algebraic variables** and associated **algebraic equations**.

System Dynamics (2/2)

- **Explicit ordinary differential equation (ODE)** system:

$$\boxed{\dot{x} = f(t, x, u) \qquad \begin{array}{l} t - \text{time} \\ x - \text{states} \\ u - \text{inputs} \end{array}}$$

- **Differential-algebraic equation (DAE) system**, fully implicit to the left and semi-explicit form to the right:

$$\begin{aligned} F(t, \dot{x}, x, z, u) &= 0, \\ G(t, x, z, u) &= 0 \end{aligned} \qquad \begin{aligned} \dot{x} &= F(t, x, z, u), \\ 0 &= G(t, x, z, u) \end{aligned} \qquad \begin{array}{l} t - \text{time} \\ x - \text{states} \\ z - \text{algebraic variables} \\ u - \text{inputs} \end{array}$$

Objective Function

- The function to be optimized (i.e., minimized or maximized) is referred to as the **objective function**.
- Also known as **cost function**, if to be minimized.
- If objective function independent of optimization variables: **feasibility problem**.
- Can involve optimization variables at any time point in the considered interval $[0, T]$, e.g., **integral of quadratic function** and **penalty on terminal states** (at time T).

Constraints

- In addition to the motion equations, there are often several other **limitations** or **requirements** to consider.
- Mathematically described by **equalities** or **inequalities**.
- Examples:
  - **Limits on control signals**, such as maximum acceleration.
  - Geometric constraints for **vehicle obstacle avoidance**.
  - **Physical constraints on internal states** and other variables, such as maximum power from motor.

A First Optimization Example (1/3)

- Assume an **autonomous car** that should move from point A to point B, driving along a straight road.
- Perform this task in as **short time as possible**.
- The quantity that we can control on the car is the **acceleration**.
- Limitations on **maximum acceleration and maximum velocity** of the car.

A First Optimization Example (2/3)

- **Mathematical formulation** of the motion-planning problem for the autonomous car over time horizon $[0, T]$:

Objective function

System dynamics

Initial conditions

Input and state constraints

Terminal constraints

$$\begin{aligned} \text{minimize} \quad & T \\ \text{subject to} \quad & \dot{p} = v, \; \dot{v} = a, \\ & p(0) = 0, \; v(0) = 0, \\ & p(T) = 100, \; v(T) = 5, \\ & a_{\min} \leq a \leq a_{\max} \\ & v_{\min} \leq v \leq v_{\max} \end{aligned}$$

p – position
v – velocity
a – acceleration
T – terminal time

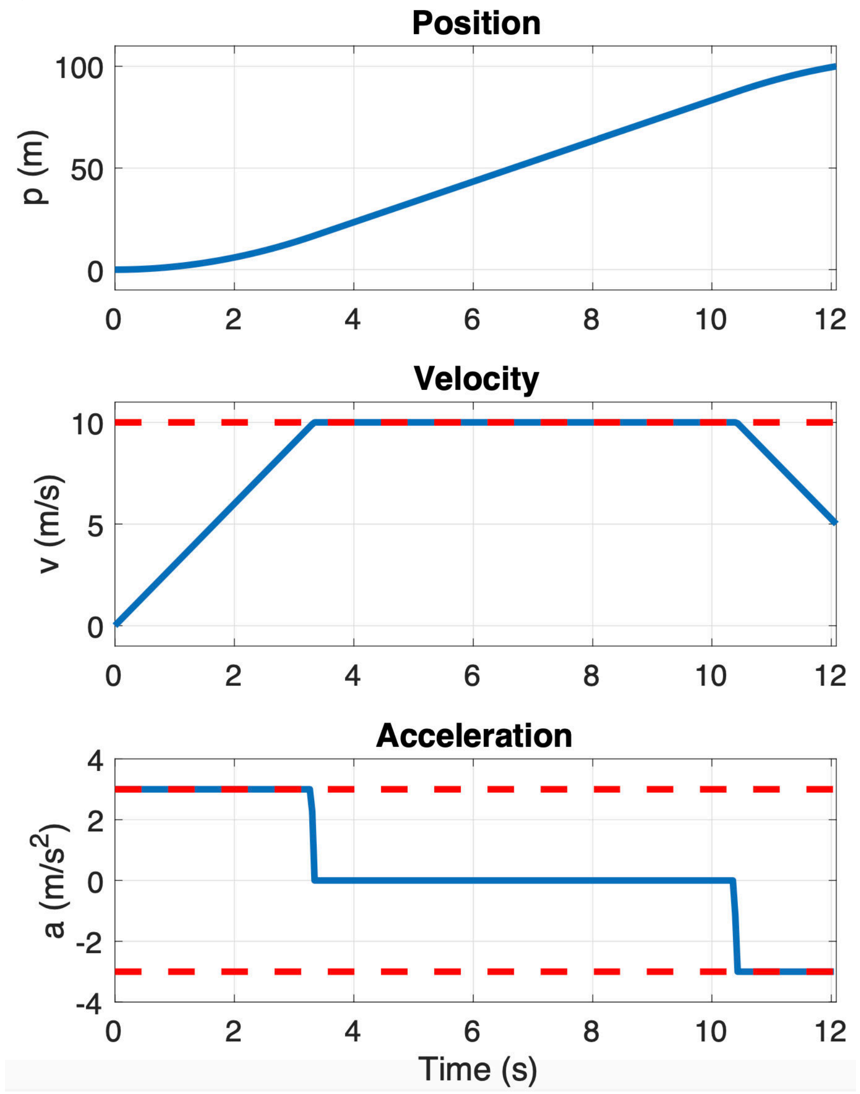
- Identification of variables leads to the expressions:

$$x = \begin{pmatrix} p & v \end{pmatrix}^{\mathrm{T}}, \; u = a, \; f = \begin{pmatrix} v & u \end{pmatrix}^{\mathrm{T}}$$

A First Optimization Example (3/3)

- Resulting **trajectories** for the **position**, **velocity**, and **acceleration** of the car.

$$
\begin{aligned}
p(0) &= 0,\ p(T) = 100\ \mathrm{m}, \\
v(0) &= 0,\ v(T) = 5\ \mathrm{m/s}, \\
v_{\min} &= -10\ \mathrm{m/s}, \\
v_{\max} &= 10\ \mathrm{m/s}, \\
a_{\min} &= -3\ \mathrm{m/s}^2, \\
a_{\max} &= 3\ \mathrm{m/s}^2
\end{aligned}
$$



Mathematical Formulation

Lagrange integrand

Mayer term

- **Optimization problem** over time horizon $[0, T]$, where T possibly is a free optimization variable:

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

Initial conditions

System dynamics

Terminal constraints

- Minimization of **terminal time** can be formulated:

State and control constraints

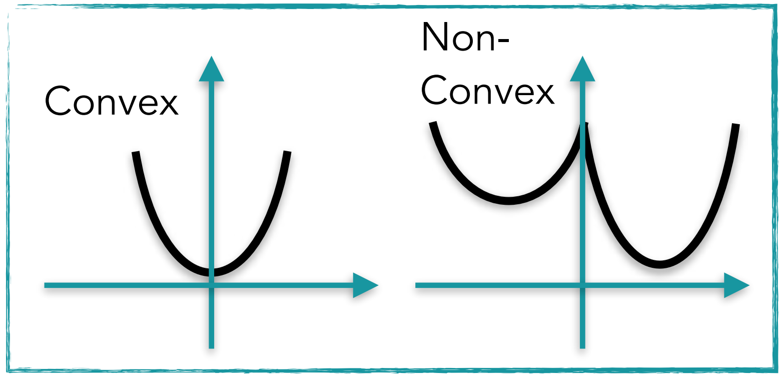
$$T = \int_0^T 1\,\mathrm{d}t$$

LINKÖPING UNIVERSITY

Challenges and Solution Strategies for Dynamic Optimization

Challenges (1/2)

- Objective function with equality and inequality constraints.
- Optimization algorithm often finds **loopholes in model**.
- Continuous-time dynamics (**infinite dimensional**).
- Often **non-linear** and **non-convex** problems in applications.



"In constrained as well as in unconstrained minimization, convexity is a watershed concept. The distinction between problems of 'convex' and 'nonconvex' type is much more significant in optimization than that between problems of 'linear' and 'nonlinear' type." – R. T. Rockafellar, Fundamentals of Optimization, Univ. of Washington, Seattle.

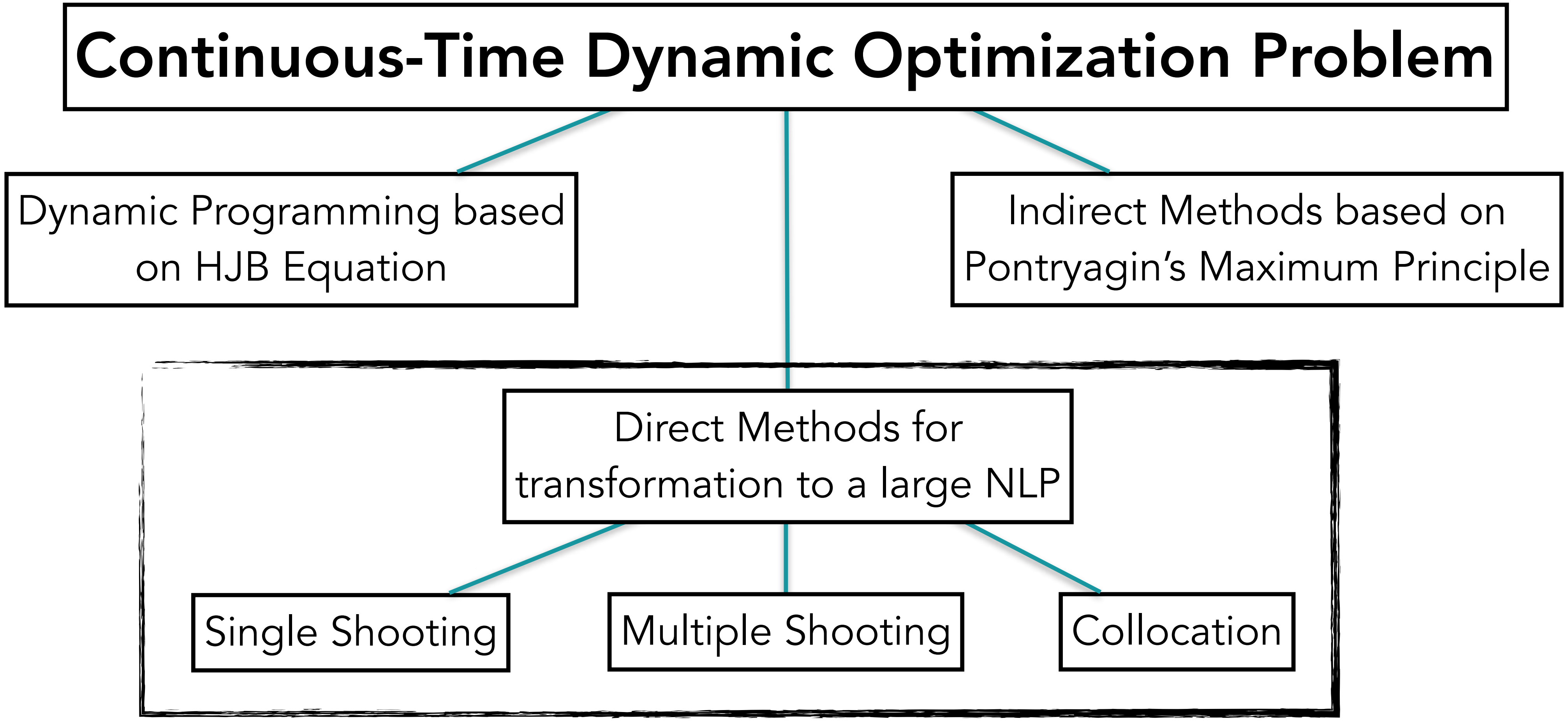
- **Local** and **global optima** of the optimization problem.

Challenges (2/2)

- **Convex optimization problem** (convex feasible set and convex objective function): *a local minimum is also a global minimum*.
- **Non-convex optimization problem**: can be challenging to even find one local optimum, and no information about objective value at possible other local optima.
- **How to find local minima**?
  - Recall from previous courses in calculus that **local optima** of a function can be found using the derivative.
  - Here we mainly consider iterations based on **Newton's method** and **optimality conditions**.
  - How to numerically compute required **derivatives** of involved functions with sufficient accuracy?

Boyd, S. & L. Vandenberghe: Convex Optimization. Cambridge University Press, 2004.

Solution Strategies – Overview


Continuous-Time Dynamic Optimization Problem
Dynamic Programming based on HJB Equation
Indirect Methods based on Pontryagin's Maximum Principle
Direct Methods for transformation to a large NLP
Single Shooting
Multiple Shooting
Collocation

Numerical Solution of an Optimal Control Problem

- Two major approaches to **discretization** related to 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 for Dynamic Optimization

Direct Methods – The Overall Idea

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

Infinite-dimensional optimal control problem

$$\Big\downarrow \ \textbf{Discretization}$$

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

Optimization problem with a finite set of variables, a non-linear program

Diehl, M: Numerical Optimal Control, Optec, K.U. Leuven, Belgium, 2011.

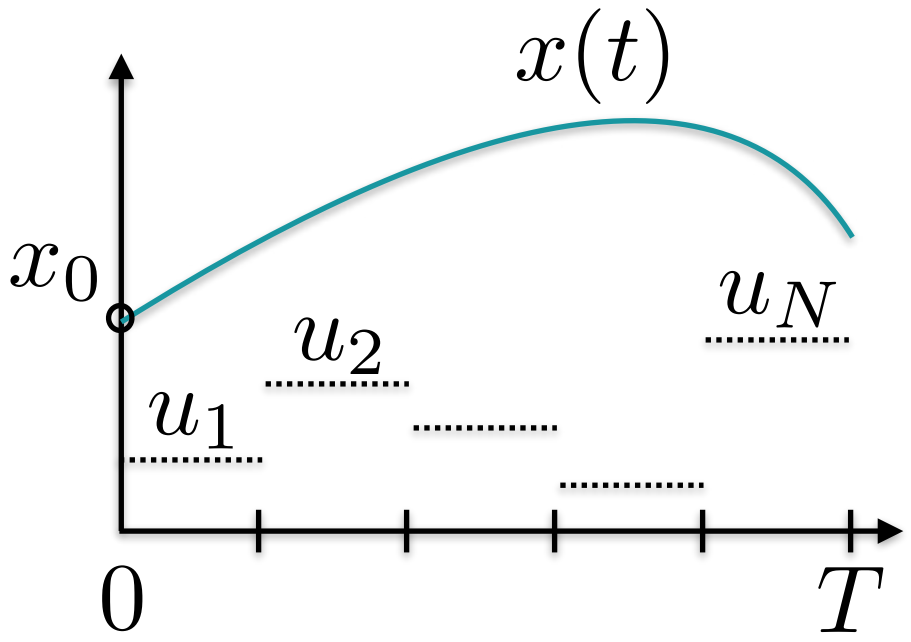
Direct Methods

- **First discretize** the optimization problem, **then optimize** to find an **approximate solution** to the original continuos-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).

Diehl, M: Numerical Optimal Control, Optec, K.U. Leuven, Belgium, 2011.

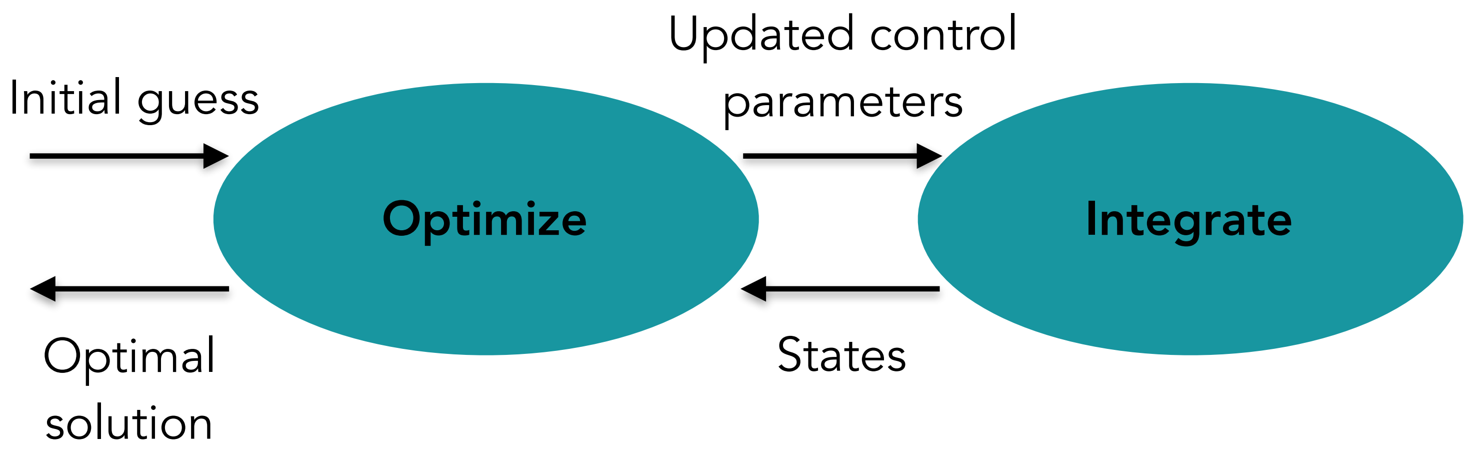
Direct Single Shooting (1/2)

- 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.



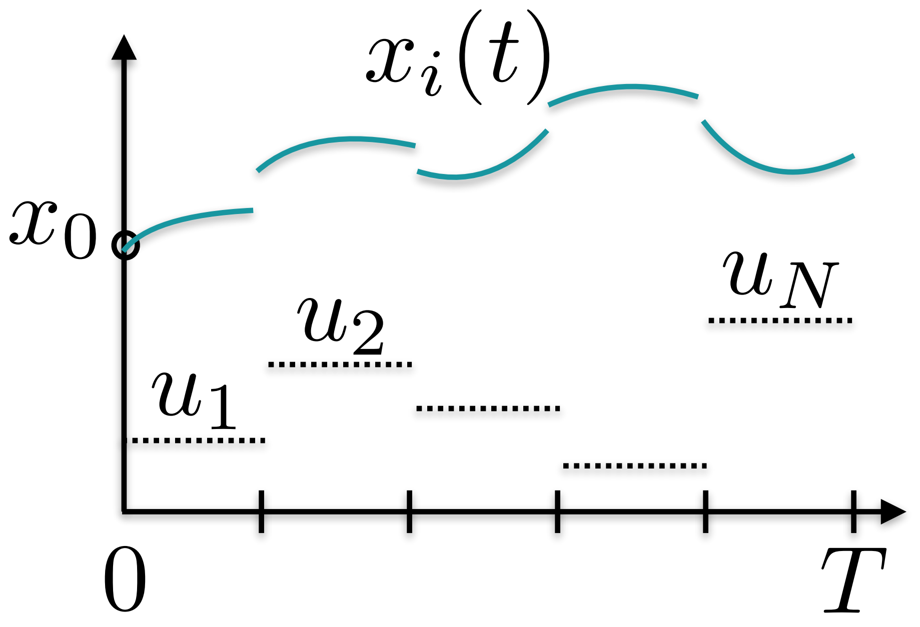
Direct Single Shooting (2/2)

- **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.



Direct Multiple Shooting (1/2)

- Extension of single shooting:
    - Divide the time horizon into **elements** and apply single shooting in each element.
    - Add **continuity constraints** (equality) at the element junctions for the states.
    - A **hybrid** between sequential and simultaneous approach.



Direct Multiple Shooting (2/2)

- **Explicit Runge-Kutta** (RK) methods common for the integration of the states in each element.
- The division into elements implies better **numerical accuracy** (especially for unstable system dynamics).
- Allows **initialization** of state trajectories and easier handling of **path constraints**.
- **Sparsity** in the Jacobian and Hessian matrices corresponding to the resulting NLP.

Rawlings, J. B., D. Q. Mayne, & M. Diehl: Model Predictive Control: Theory, Computation, and Design. 2nd Edition. Nob Hill Publishing, 2017.

Direct Simultaneous Collocation (1/4)

- 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**.

Explicit

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

Implicit

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

Direct Simultaneous Collocation (2/4)

- 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**), often using implicit integration methods.
- Define a number of **collocation points** within each element.

Direct Simultaneous Collocation (3/4)

- Represent state variables within each element using **Lagrange interpolation polynomials** (*piecewise polynomials* over the time horizon) of order N_c .
- **Collocation polynomials** for state derivatives obtained by differentiating Lagrange polynomials for the state variables.
- Add **state-boundary constraints** at the element junctions for continuity.

Direct Simultaneous Collocation (4/4)

- Choice of collocation points within each element leads to different versions of *implicit Runge-Kutta methods* (nonlinear equation system):
  - **Gauss-Legendre** – collocation points chosen as zeros of orthogonal Legendre polynomials, strictly in interior of element and symmetric around midpoint of element.
  - **Radau** – always includes the end point of the element, provides so called stiff decay and exhibits good numerical stability in many applications.
  - **Lobatto** – always includes the initial point and the end point of the element as collocation points.
- Variant of direct collocation: **pseudo-spectral methods** where only one element over the entire time horizon $[0, T]$, but high-order polynomials for the interpolation polynomials (i.e., many collocations points).

Magnusson, F: Numerical and Symbolic Methods for Dynamic Optimization, Ph.D. Thesis TFRT-1115, Dept. Automatic Control, Lund University, 2016.

Collocation – Example (1/2)

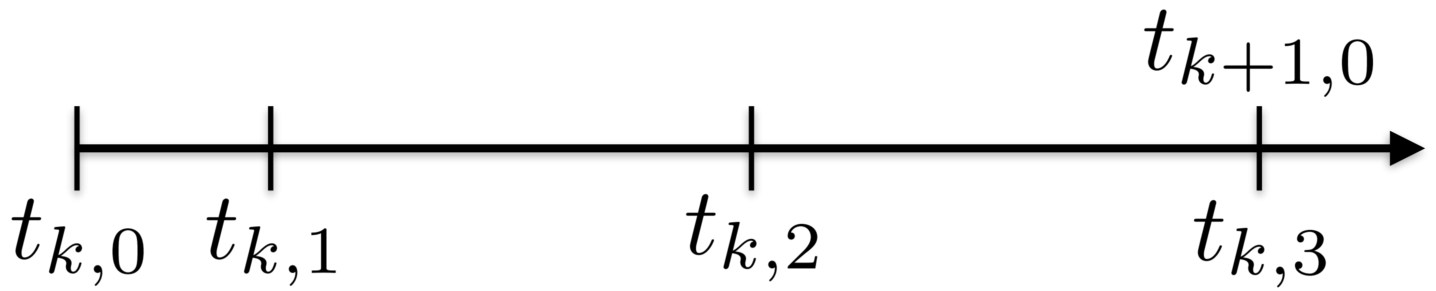
- Approximate the state trajectories with **piecewise third-order polynomials** and collocation points using **Radau scheme**.
- Points distributed in the normalized interval $[0, 1]$:

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

- Introduce a uniform width of each element

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

- Collocation points illustrated graphically (non-uniform!)



Collocation – Example (2/2)

- Within each element, **Lagrange polynomials** basis are 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

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

- **Differentiation** with respect to time gives

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

Solution of the Resulting Non-Linear Program

Non-Linear Program (NLP)

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

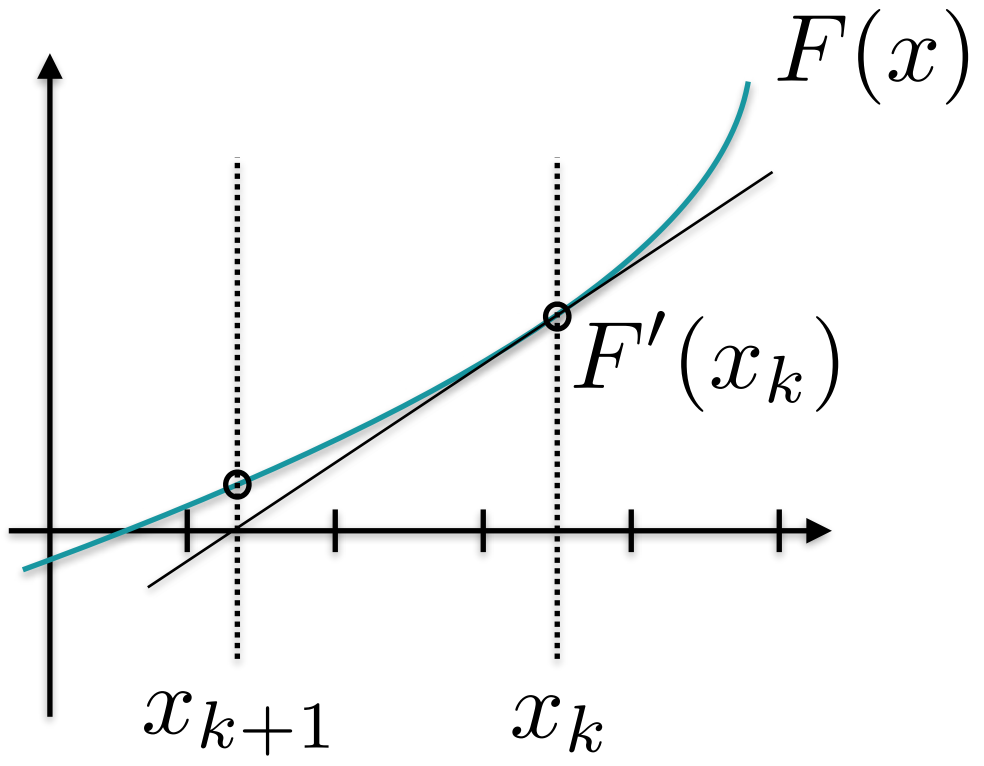
$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & g(x) = 0, \\ & h(x) \leq 0 \end{aligned}$$

- This NLP can be solved using various methods, e.g., **interior point** (IP) and **sequential quadratic programming** (SQP).

Nocedal, J., & S. Wright: Numerical Optimization. Springer, 2006.

Background: 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}.



Background: Newton's Method (2/2)

- **Iteratively linearize** the function F around the current x using the Jacobian J and 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**:

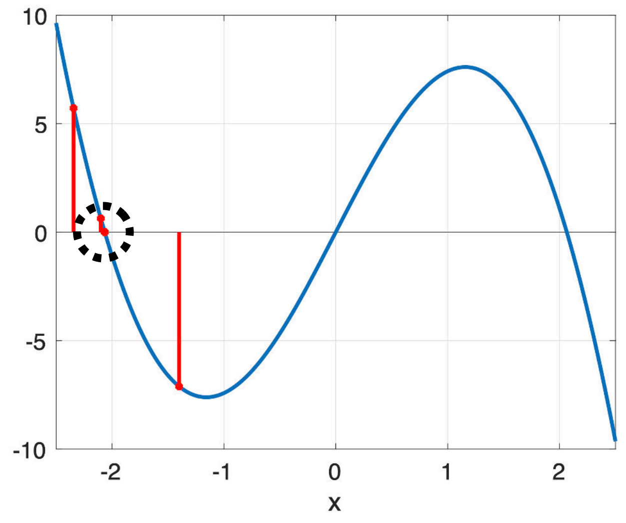
$$J(x_k)\Delta x = -F(x_k), \text{ where } \Delta x = x_{k+1} - x_k$$

- Can be used for finding **local optimum** for optimization problem – **initialization strategies** for variables important.

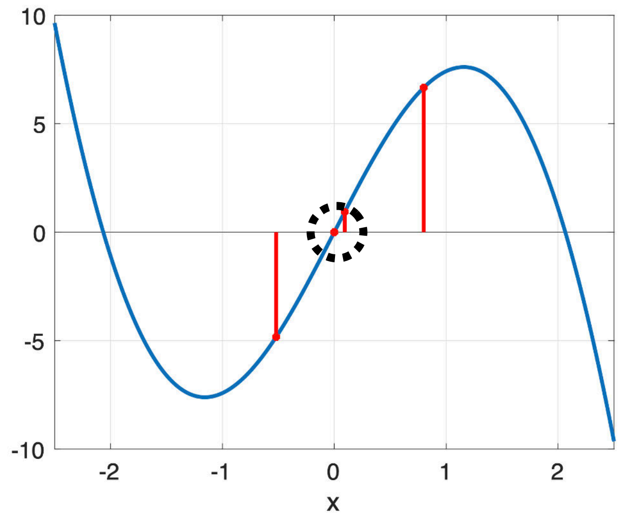
Newton's Method – Example

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

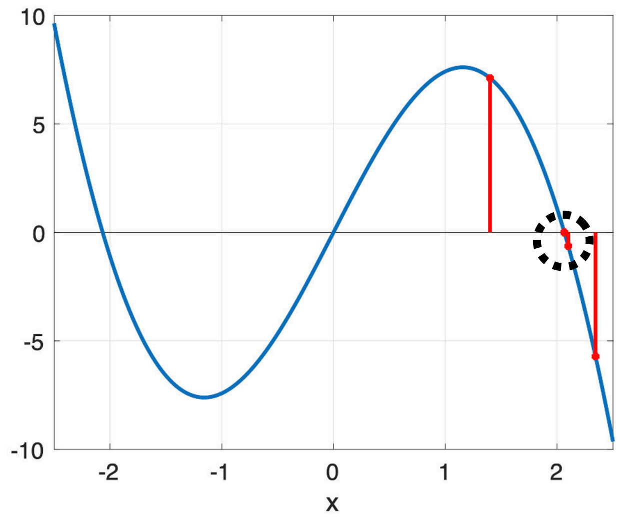
$$F(x) = 10\sin(x) - x^3, \quad F'(x) = 10\cos(x) - 3x^2$$



$x_0 = -1.4$



$x_0 = 0.8$



$x_0 = 1.4$

Automatic Differentiation – AD

- Structured way of **computing derivatives** with machine precision.
- *Chain rule* for differentiation used to **decompose** the problem into smaller **elementary operations**.
- Provides Jacobians (first-order derivatives) and Hessians (second-order derivatives) for solution of the NLP 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.

Nocedal, J., & S. Wright: Numerical Optimization. Springer, 2006.

Automatic Differentiation – Example

- Compute gradient of F using **AD in forward mode**:

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

- Decompose into **elementary operations**:

$$x_3 = x_1 x_2, \qquad \dot{x}_3 = \dot{x}_1 x_2 + x_1 \dot{x}_2,$$

$$x_4 = \cos(x_1), \qquad \dot{x}_4 = -\sin(x_1)\dot{x}_1,$$

$$x_5 = x_3 + x_4 \qquad \dot{x}_5 = \dot{x}_3 + \dot{x}_4$$

- 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.

Optimality Conditions

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

- Introduce the **Lagrangian function**:

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

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

$$\begin{aligned} \nabla_x \mathcal{L}(x^*, \lambda^*, \nu^*) &= 0, && \leftarrow \text{Stationarity} \\ g(x^*) &= 0, && \leftarrow \\ h(x^*) &\leq 0, && \leftarrow \text{Feasibility} \\ \nu^* &\geq 0, && \leftarrow \text{Complementarity} \\ \nu_i^* h_i(x^*) &= 0, \ \ i = 1, \ldots, n_h && \leftarrow \end{aligned}$$

Solution of NLP (1/2)

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g(x) = 0, \\ & h(x) \le 0 \end{array}$$

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

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

- Apply **Newton's method** with variables and function:

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

Solution of NLP (2/2)

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

- Application of Newton's method for the case $h(x) = 0$ gives the iterations as the solution of the **linear equation system**:

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

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

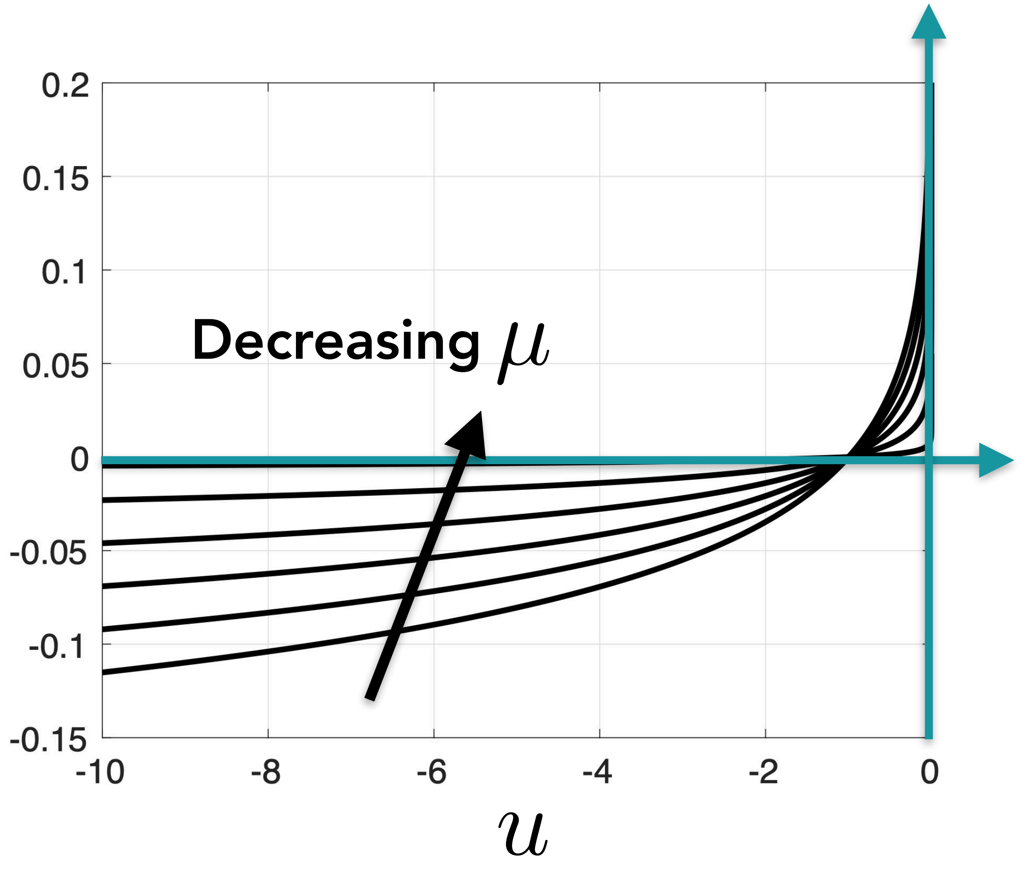
Nocedal, J., & S. Wright: Numerical Optimization. Springer, 2006.

Example: Logarithmic Barrier Function

- Consider the following **approximation** of a **barrier function**:

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

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



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

Interior-Point Methods (1/2)

- Consider **equality and inequality-constrained NLP** problems.
- **Interior-point methods** with **barrier functions for inequalities** – move inequality constraints to objective function with barrier function and positive parameter μ:

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

- Could then be solved as an **equality-constrained problem**, but often Lagrange variables for inequalities kept for numerical stability as shown on next slide.

Nocedal, J., & S. Wright: Numerical Optimization. Springer, 2006.

Interior-Point Methods (2/2)

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

- Log-barrier approach corresponds to a **smooth approximation** of the KKT system:

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

- **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.

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.

Sequential Quadratic Programming

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

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

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

- QP can be solved using **IP methods** or **active set methods**.

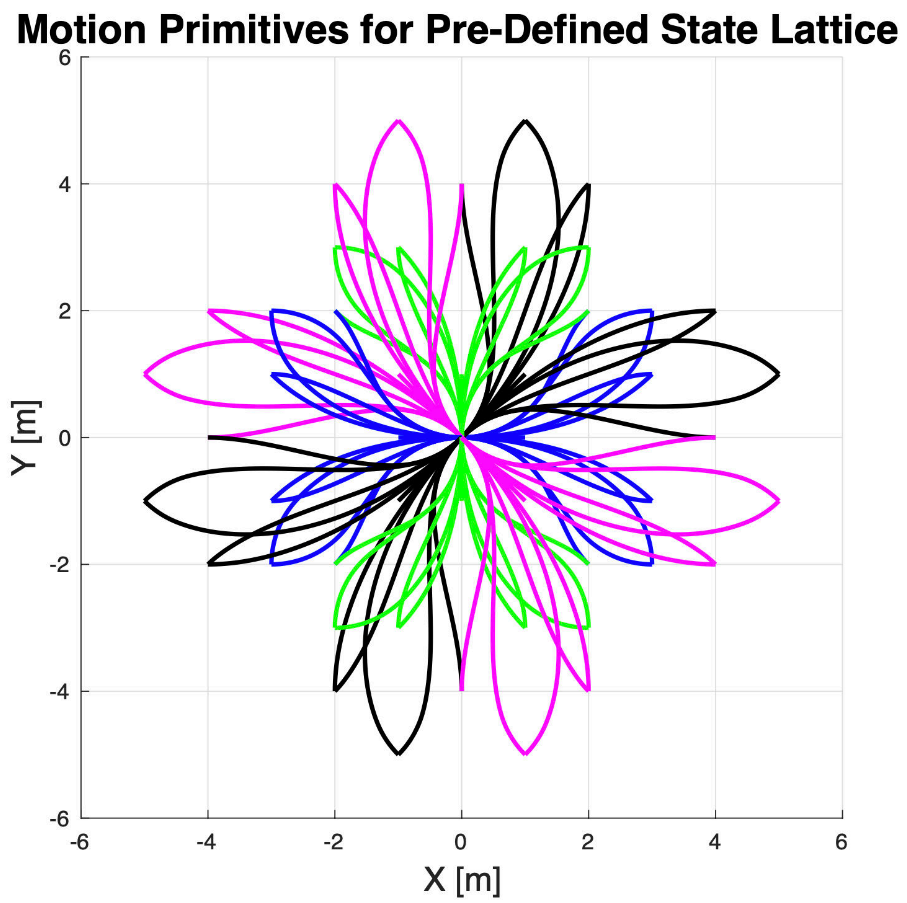
Nocedal, J., & S. Wright: Numerical Optimization. Springer, 2006.
Diehl, M.: Numerical Optimal Control, Optec, K.U. Leuven, Belgium, 2011.

Case Study on Dynamic Optimization

Case Study: Optimization of Motion Primitives (1/5)

- **Optimization** can be used to compute the **motion primitives** from Lecture 4 (see also Hand-in Exercise 2).
- Example code in Matlab using the tool CasADi on the following slides.
- Vehicle motion equations given by:

$$\dot{x} = v\cos(\theta),$$
$$\dot{y} = v\sin(\theta),$$
$$\dot{\theta} = v/L\tan(u)$$



Case Study: Optimization of Motion Primitives (2/5)

```
% Compute the motion primitive using optimization with the tool CasADi
% using direct collocation for discretization of the continuous-time
% motion equations.

% Parameters for collocation
N = 75; % Number of elements
nx = 3; % Degree of state vector
Nc = 3; % Degree of interpolation polynomials

x_vec = lattice(1, :);
y_vec = lattice(2, :);
th_vec = lattice(3, :);

% Formulate the optimization problem for minimum path length using CasADi

import casadi.*

for i = 1:length(x_vec)

    % Use the opti interface in CasADi
    opti = casadi.Opti();
```

Case Study: Optimization of Motion Primitives (3/5)

```
state_f = [x_vec(i) y_vec(i) th_vec(i)]';

% Define optimization variables and motion equations
x = MX.sym('x',nx);
u = MX.sym('u');

f = Function('f',{x, u}, {v*cos(x(3)), v*sin(x(3)), v*tan(u)/L});
X = opti.variable(nx,N+1);
pos_x = X(1, :);
pos_y = X(2, :);
ang_th = X(3, :);

U = opti.variable(N, 1);
T = opti.variable(1);

% Set the element length (with final time T unknown, and thus an
% optimization variable)
dt = T/N;

% Set initial guess values of variables
opti.set_initial(T, 0.1);
opti.set_initial(U, 0.0*ones(N, 1));
opti.set_initial(pos_x, linspace(state_i(1), state_f(1), N+1));
opti.set_initial(pos_y, linspace(state_i(2), state_f(2), N+1));
```

Case Study: Optimization of Motion Primitives (4/5)

```
% Define collocation parameters
tau = collocation_points(Nc, 'radau');
[C,~] = collocation_interpolators(tau);

% Formulate collocation constraints

for k = 1:N   % Loop over elements
    Xc = opti.variable(nx, Nc);
    X_kc = [X(:, k) Xc];
    for j = 1:Nc
        % Make sure that the motion equations are satisfied at
        % all collocation points
        [f_1, f_2, f_3] = f(Xc(:, j), U(k));
        opti.subject_to(X_kc*C{j+1}' == dt*[f_1; f_2; f_3]);
    end
    % Continuity constraints for states between elements
    opti.subject_to(X_kc(:, Nc+1) == X(:, k+1));
end
```

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

```
% Input constraints
for k = 1:N
    opti.subject_to(-u_max <= U(k) <= u_max);
end
```

Case Study: Optimization of Motion Primitives (5/5)

```
    % Initial and terminal constraints
    opti.subject_to(T >= 0.001);
    opti.subject_to(X(:, 1) == state_i);
    opti.subject_to(X(:, end) == state_f);

    % Formulate the cost function
    alpha = 1e-2;
    opti.minimize(T + alpha*sumsqr(U));

    % Choose solver ipopt and solve the problem
    opti.solver('ipopt', struct('expand', true), struct('tol', 1e-8));
    sol = opti.solve();

    % Extract solution trajectories and store them in the mprim variable
    pos_x_opt = sol.value(pos_x);
    pos_y_opt = sol.value(pos_y);
    ang_th_opt = sol.value(ang_th);
    u_opt = sol.value(U);
    T_opt = sol.value(T);

    mprim{i}.x = pos_x_opt;
    mprim{i}.y = pos_y_opt;
    mprim{i}.th = ang_th_opt;
    mprim{i}.u = u_opt;
    mprim{i}.T = T_opt;
    mprim{i}.ds = T_opt*v;
end
```

References and Further Reading

LINKÖPING UNIVERSITY

References and Further Reading (1/2)

All the following books and articles are not part of the reading assignments for the course, but cover the topics studied during this lecture in more detail.

- 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 ”, Ph.D. Thesis No. 2133, Div. Automatic Control, Linköping Univ., 2021.
- Berntorp, K., B. Olofsson, K. Lundahl, & L. Nielsen: ”Models and methodology for optimal trajectory generation in safety-critical road-vehicle manoeuvres”, Vehicle System Dynamics, 52(10):1304–1332, 2014.
- Boyd, S. & L. Vandenberghe: *Convex Optimization*. Cambridge University Press, 2004.
- Diehl, M: *Numerical Optimal Control*, Optec, K.U. Leuven, Belgium, 2011.
- Limebeer, D. J., & A. V. Rao: ”Faster, higher, and greener: Vehicular optimal control”. *IEEE Control Systems Magazine*, 35(2), 36-56, 2015.

References and Further Reading (2/2)

- Magnusson, F: ”Numerical and Symbolic Methods for Dynamic Optimization”, Ph.D. Thesis TFRT-1115, Dept. Automatic Control, Lund University, 2016.
- 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.
- Åkesson, J, K.-E. Årzén, M. Gäfvert, T. Bergdahl, & H. Tummescheit: ”Modeling and Optimization with Optimica and JModelica.org–Languages and Tools for Solving Large-Scale Dynamic Optimization Problems”, *Computers and Chemical Engineering*, vol. 34, no. 11, pp. 1737-1749, 2010.

www.liu.se

LINKÖPING UNIVERSITY