Dynamic Optimization as a Tool for Motion Planning and Control

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

Lecture 5: Erik Frisk <erik.frisk@liu.se>

LINKÖPINGS UNIVERSITET

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.
- Model Predictive Control
  - Optimal path following, multi-vehicle collision avoidance, …
  - Connected to MPC lecture, hand-in 3, extra assignment.
- Optimization is a good language for merging control, learning — especially for dynamic systems

Autonomous racing @ Stanford

- Optimality
- Dynamic behavior
- Real-time

https://www.youtube.com/watch?v=joIsgP9StAY

Dynamic Optimization – Background

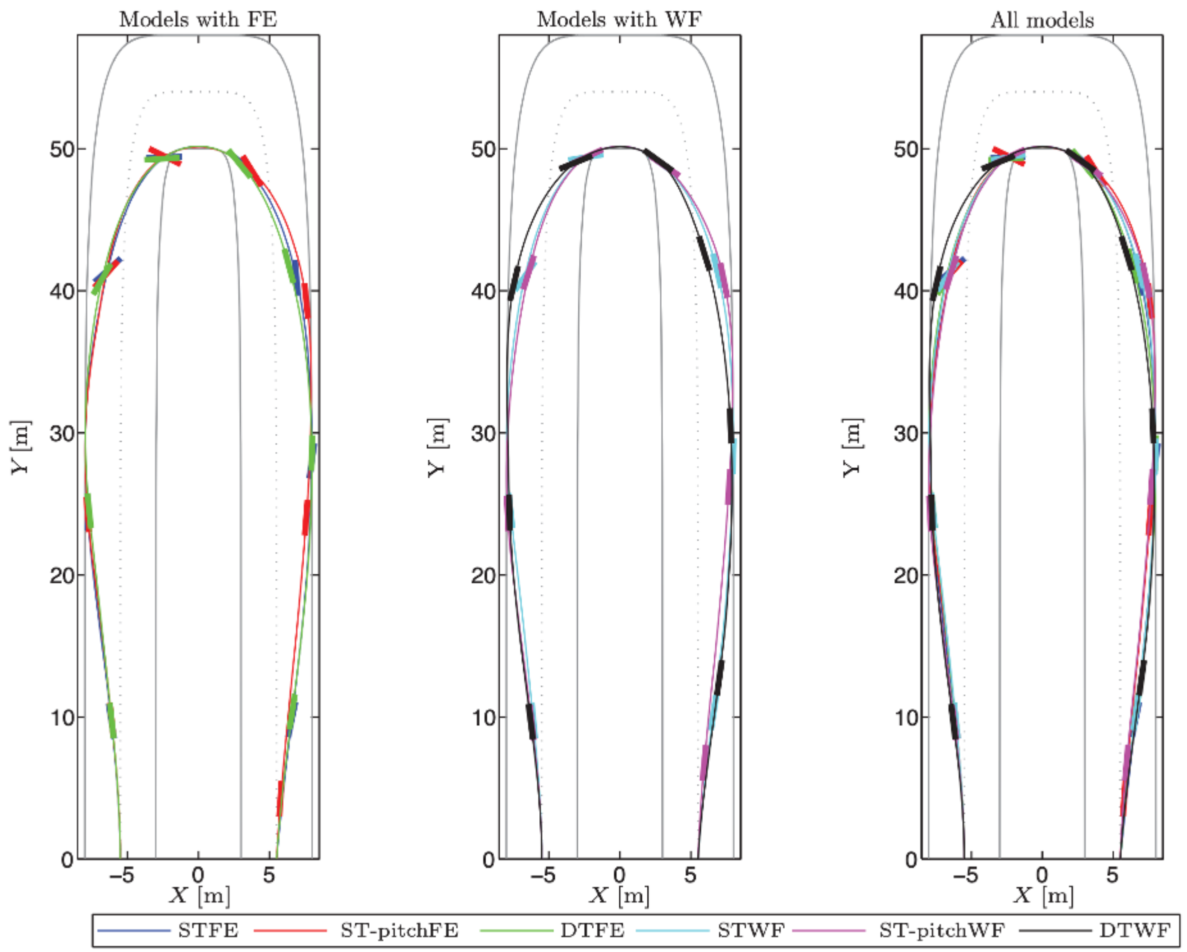
- In this lecture the focus is on numerical methods for dynamic optimization, since many real-world problems are intractable with analytical methods.
- Examples:
  - Motion planning for autonomous vehicles and robots,
  - Optimal battery-charging planning for autonomous electric vehicles,
  - Efficient electric motor and engine control,
  - Traffic-flow optimization in city environments.
  - …

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.
- Introduce, but no in-depth explanations, of some key notions
- Have knowledge about different applications of dynamic optimization for motion planning and control for autonomous vehicles.

Dynamic Optimization – Examples

- 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.



Dynamic flight

https://www.youtube.com/watch?v=MvRTALJp8DM

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.

Context in the Architecture for an Autonomous Vehicle

Dynamic optimization is possible to apply at all layers in the architecture for different tasks

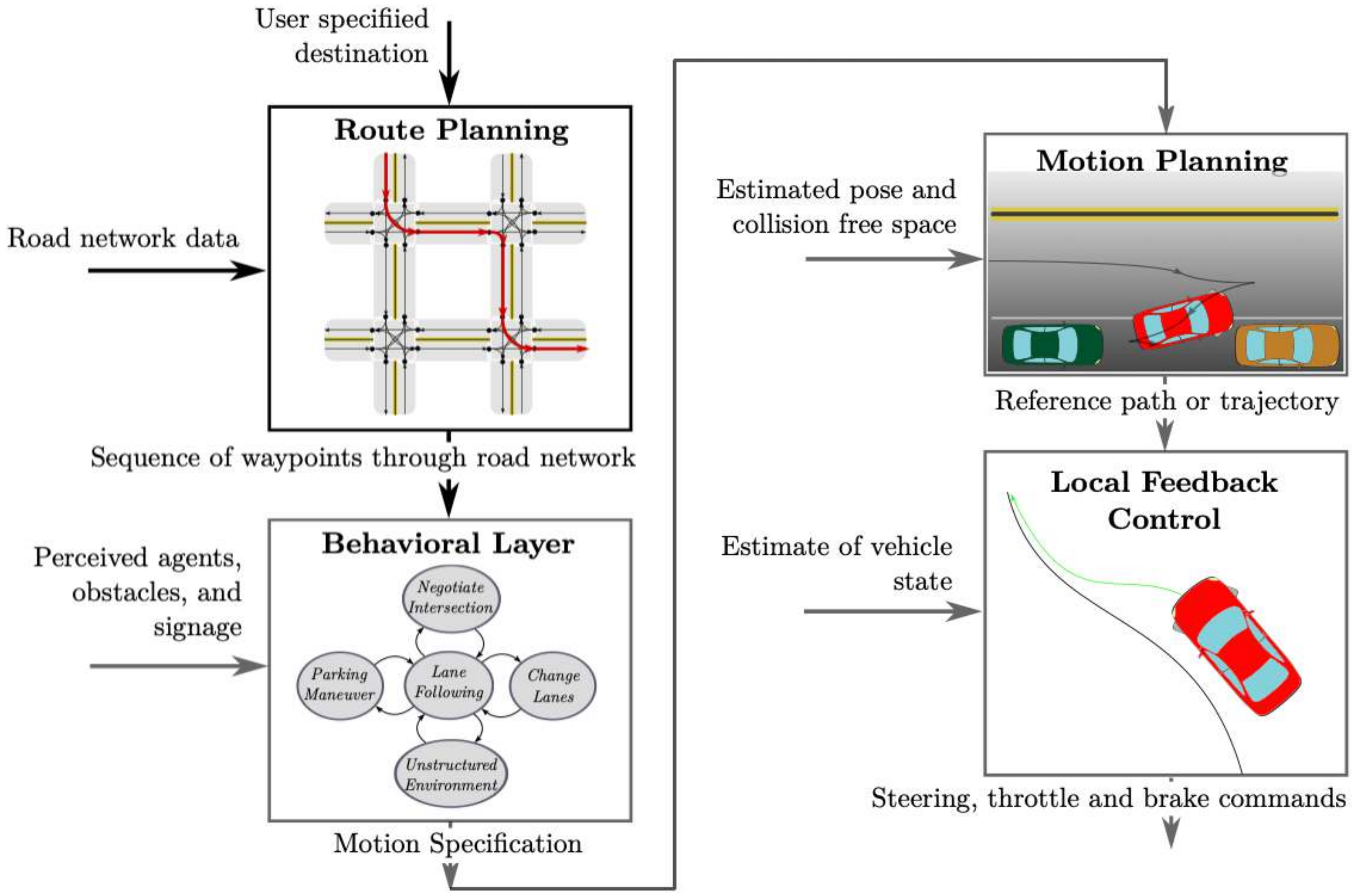
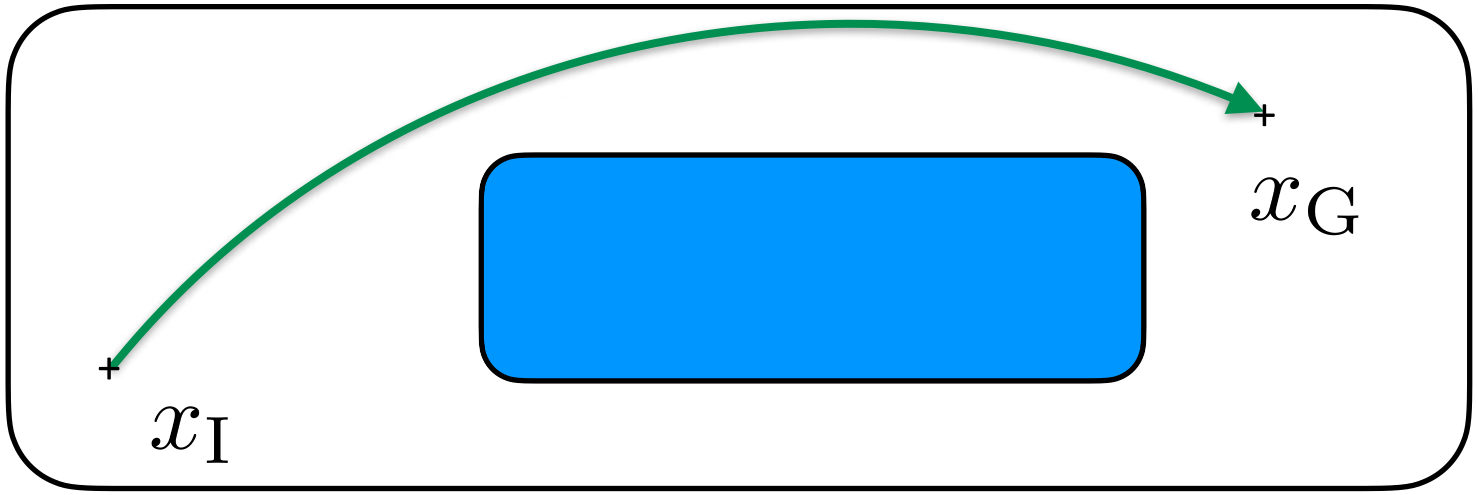

User specified destination
Route Planning
Road network data
Sequence of waypoints through road network
Behavioral Layer
Perceived agents, obstacles, and signage
Negotiate Intersection
Parking Maneuver
Lane Following
Change Lanes
Unstructured Environment
Motion Specification
Motion Planning
Estimated 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.

Formulating Dynamic Optimization Problems

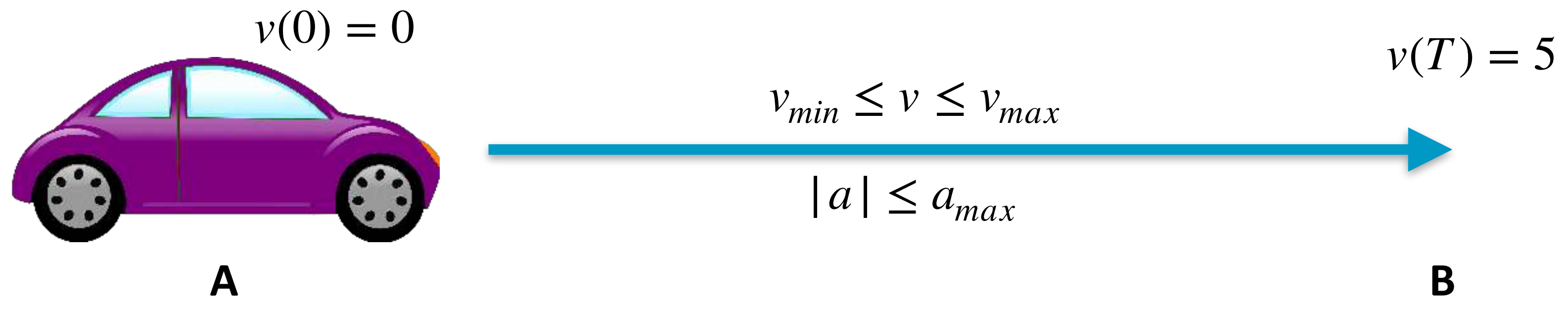
Recall Motion-Planning Problem from Lecture 4

- 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.



A First Optimization Example (1/3)

- From A to B in minimum time with velocity and acceleration constraints



- The quantity that we can control on the car is the acceleration a.
- **Solution**: accelerate as fast as possible to $v = v_{max}$ and time deceleration such that velocity at the end matches specification
- Add constraints on comfort, e.g., $|\dot{a}| < a_{max}$, then not so simple ($\dot{a} - jerk$).

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

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

Initial conditions

Terminal constraints

Input and state constraints

p – position

v – velocity

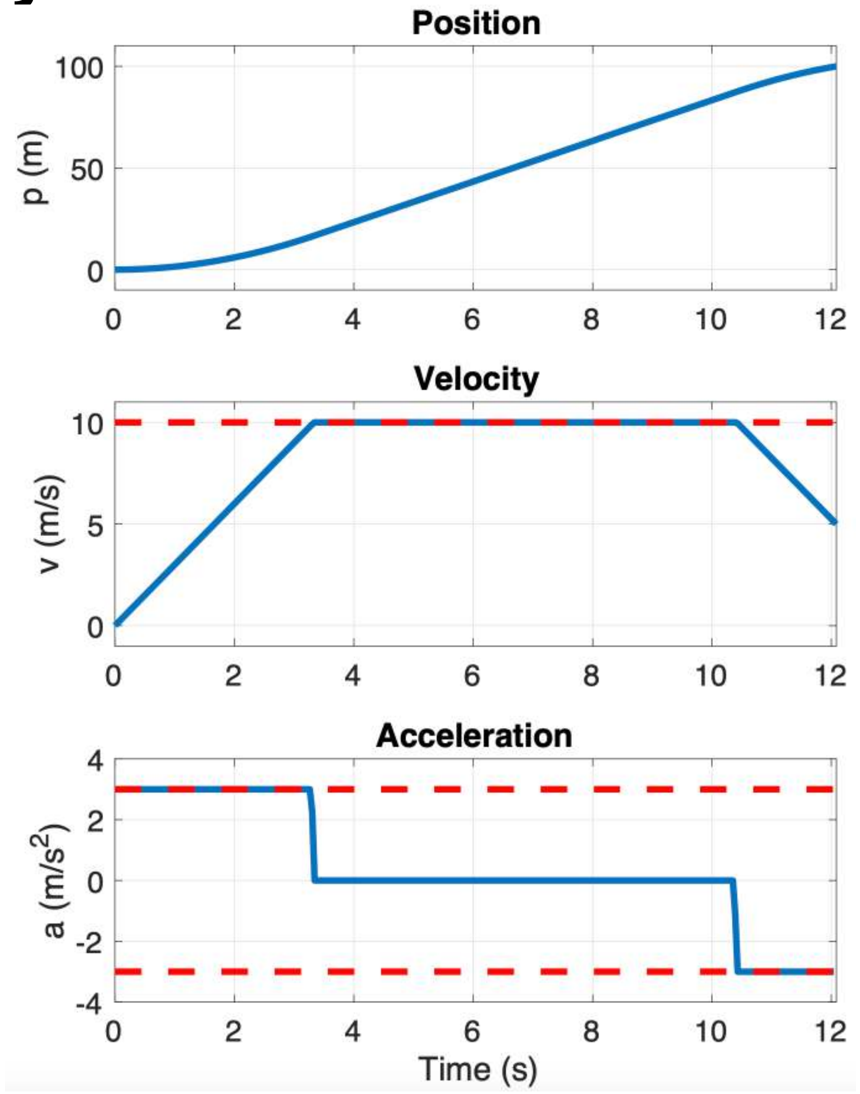
a – acceleration

T – terminal time

A First Optimization Example (3/3)

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

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



A general description – System Dynamics

- The motion equations, described by differential equations (or difference equations in discrete time), see Lecture 3.
- Often an ODE description

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

t – time
x – states
u – inputs

- It is possible to extend to formulations with algebraic variables and equations (DAE).

A general description – Objective Function

- The function to be optimized (i.e., minimized or maximized) is referred to as the objective function, loss function, cost function

$$J = \int_0^T L(x(t), u(t))\, dt + \Gamma(x(T))$$

- 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).
- Minimize time: $L(x(t), u(t)) = 1$ and $\Gamma(x(T)) = 0$, i.e.,

$$J = \min \int_0^T 1\, dt$$

A general description – Constraints

- In addition to the motion equations, there are often several other limitations or requirements to consider.
- 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.
- Mathematically described by equalities or inequalities.

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

State and control constraints

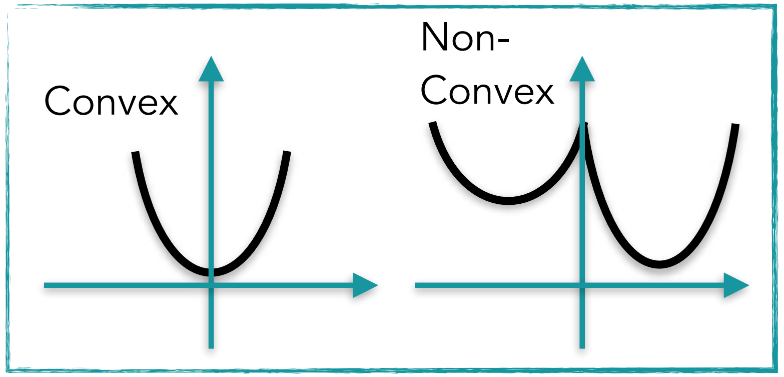
Terminal constraints

Challenges and Solution Strategies for Dynamic Optimization

LINKÖPINGS UNIVERSITET

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.

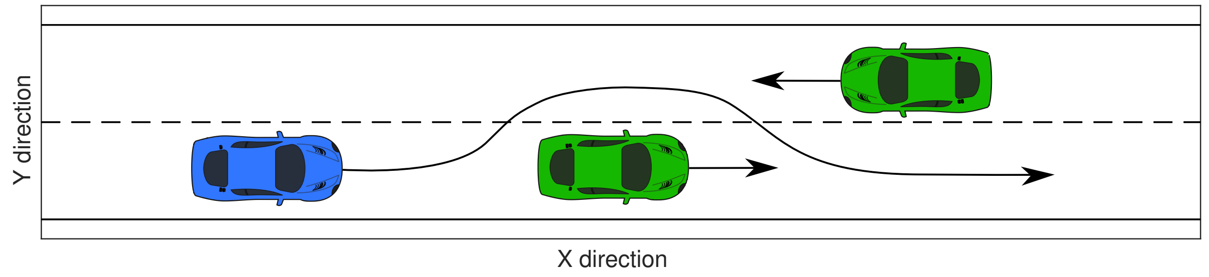


"In constrained as well as in unconstrained minimization, convexity is a watershed concept. The distinction between problems of 'convex' and 'nonconvex' type is much more significant in optimization than that between problems of 'linear' and 'nonlinear' type."
– R. T. Rockafellar, Fundamentals of Optimization,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

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

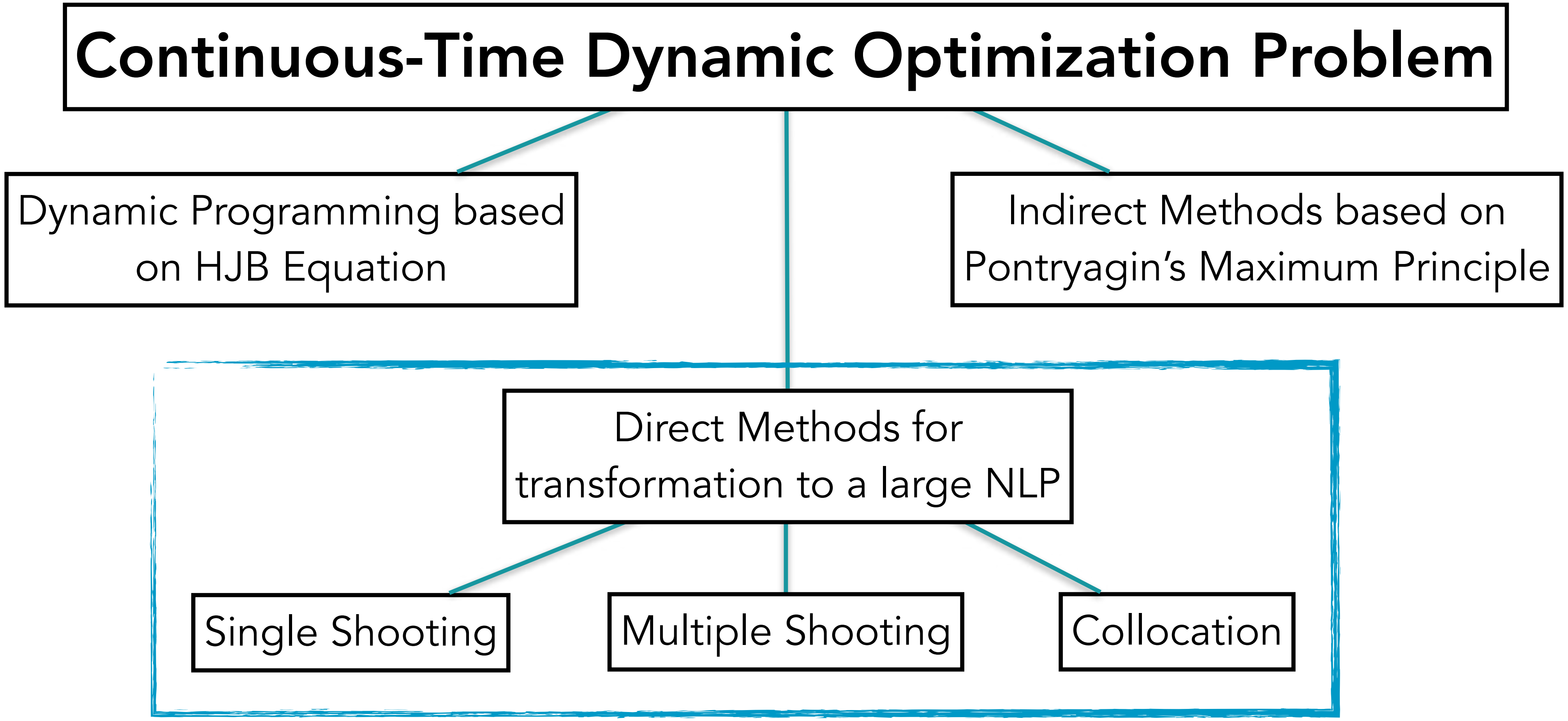
Challenges (2/2)

- **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
- Very common



- **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?

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**).

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

Direct Methods for Dynamic Optimization

— or how to discretize dynamic optimal control problems

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

$$\Big\downarrow$$

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

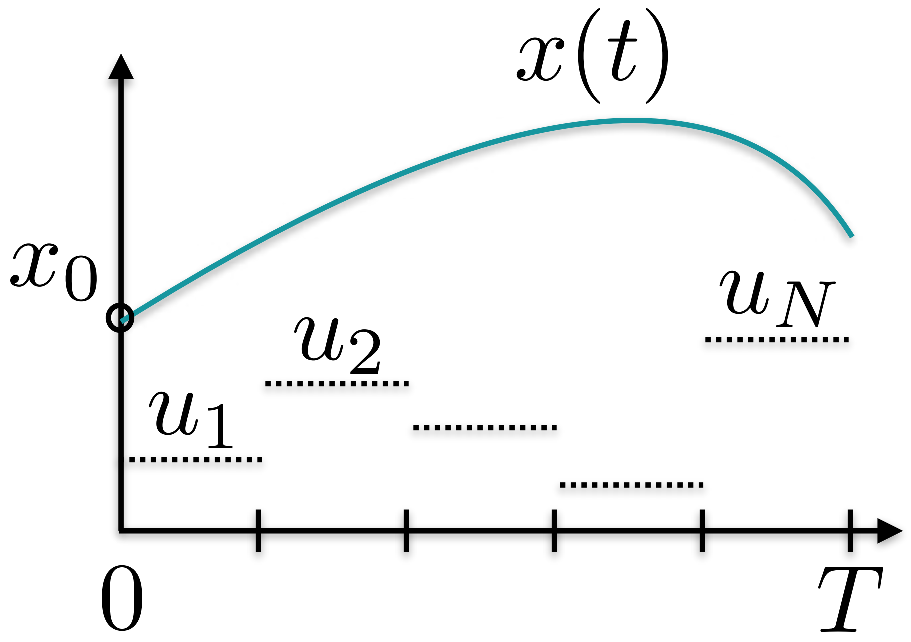
Direct Methods

- **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).

Direct Single Shooting (1/2)

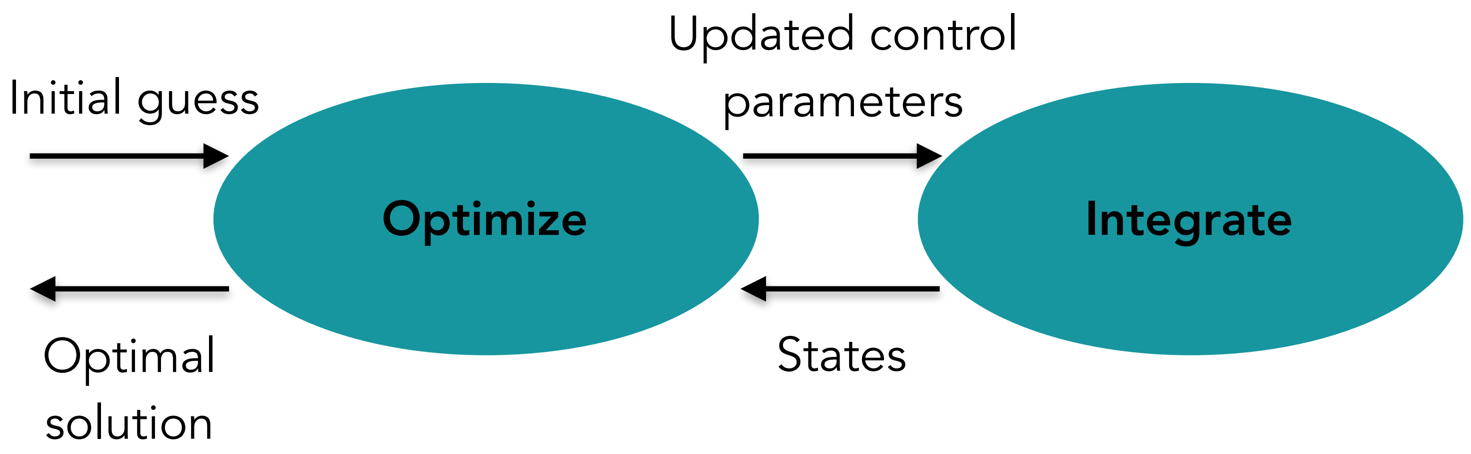
$$\begin{array}{ll} \underset{u}{\text{minimize}} & \int_0^T L(x(t),u(t))\,\mathrm{d}t + \Gamma(x(T)) \\ \text{subject to} & x(0)=x_0,\ \dot{x}(t)=f(t,x(t),u(t)) \\ & x(t)\in\mathbb{X},\ u(t)\in\mathbb{U},\ x(T)\in\mathbb{X}_T, \end{array}$$

- 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.



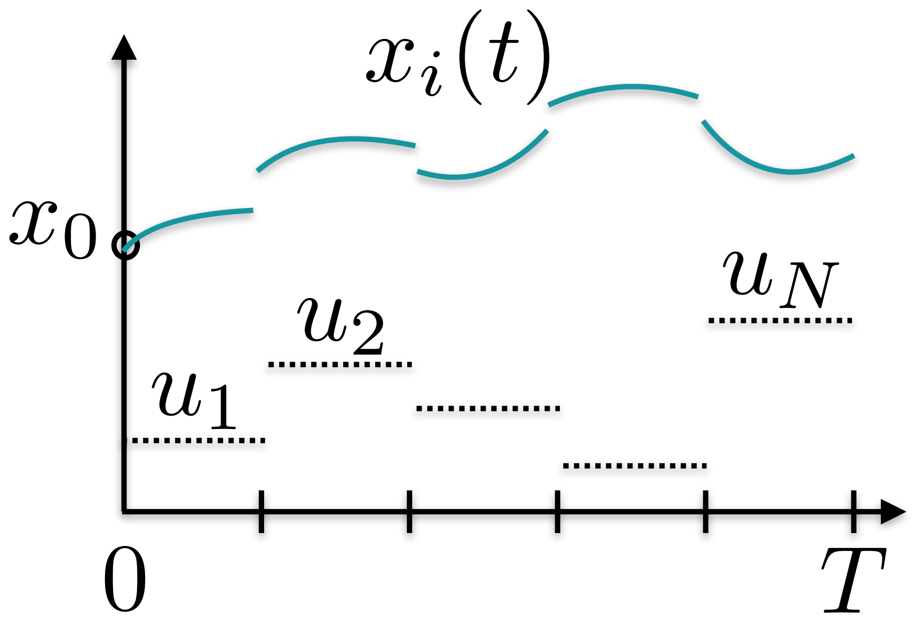
Direct Single Shooting (2/2)

- **Sequential** approach that optimizes over the control parameters (typically assuming piecewise constant control).
- One of the most straightforward methods to implement
- **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.
- Optimization over u_i and the continuity variables



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.

Direct Simultaneous Collocation *"keep the states"* (1/3)

- Consider an explicit ODE:

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

- Numerical integration with **explicit or implicit Runge-Kutta methods**
- 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 (2/3)

- 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.

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

Collocation – Example (1/2)

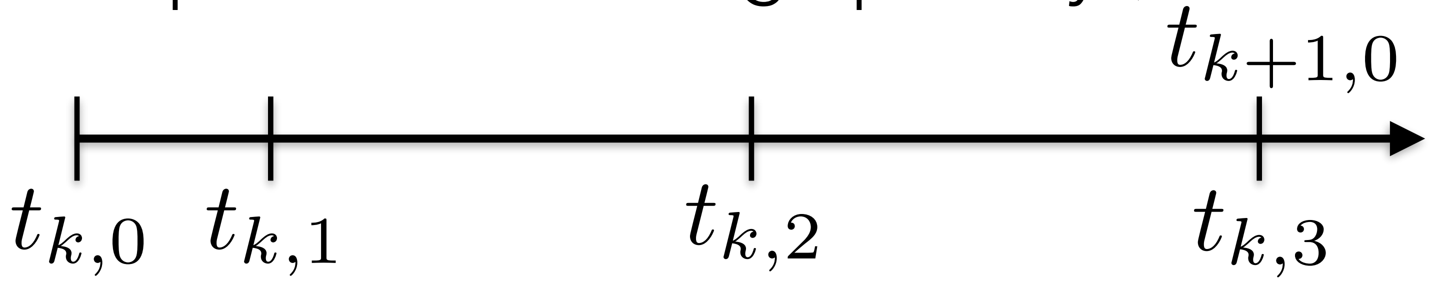
- Approximate the state trajectories with **piecewise third-order polynomials** and collocation points using **Radau scheme**.
- Points distributed in the normalized interval $[0,1]$:

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

- Introduce a uniform width of each element

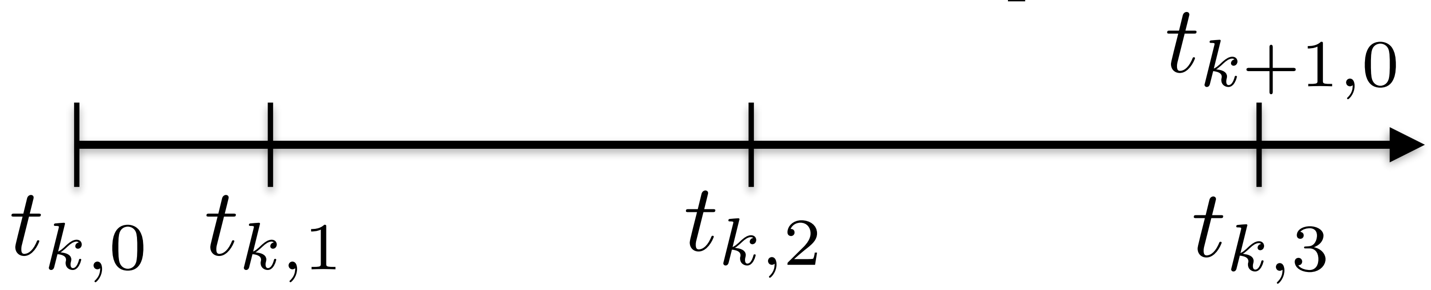
$$t_k = kh, \quad k = 0, \ldots, N$$

- Collocation points illustrated graphically (non-uniform!)



Collocation - main idea summary

- Discretize both state and input trajectories, i.e., optimize over both control u and state x
- Direct to handle control and state constraints
- Include integration technique, often a collocation method, meaning that inter step-state variables are included in the optimization



- Leads to large optimization problems with a lot of structure — exists a large body of research and tools to solve efficiently

Solution of the Resulting Non-Linear Program

a quick introduction

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.

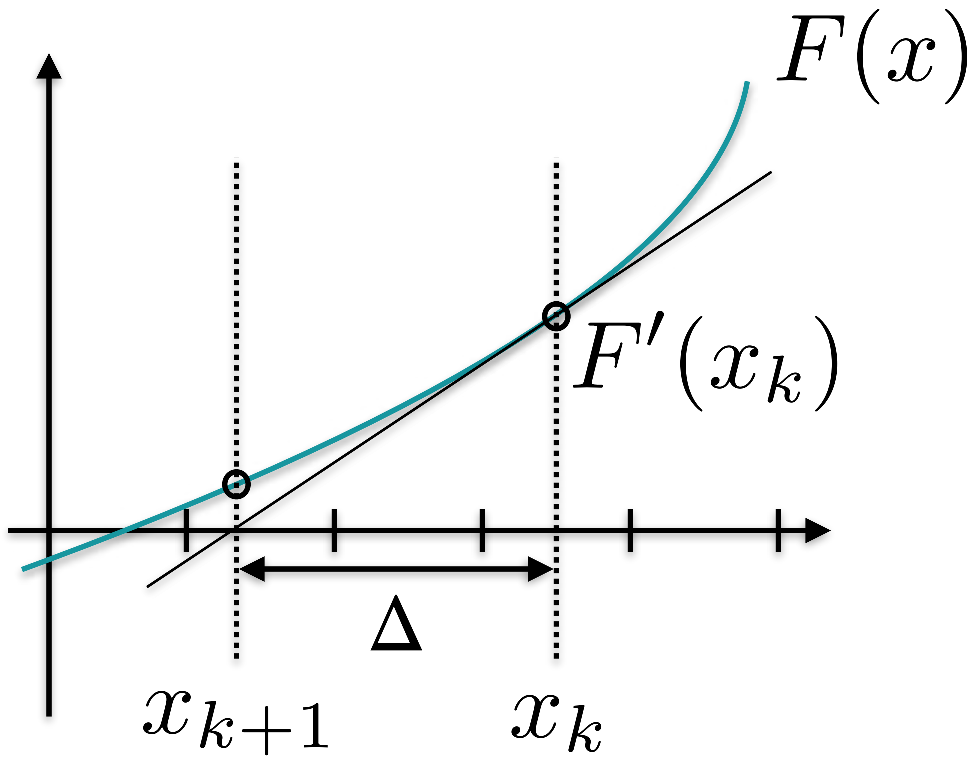
Optimization methods

- This is not a course in optimization but choosing the right solver for the right problem essential for performance
  - wrong choice can easily reduce performance orders of magnitude
- Convex vs. Non-convex problems
- Common methods in dynamic optimization
  - QP - Quadratic Program
  - SQP - Solve the optimization problem by a sequence of QP
  - Interior point methods - move constraints to the objective via barrier functions
  - … and so many more
- Take-away here: keep in mind that method matters and look it up when you need to

Background: Newton's Method

- **Iterative method** for finding a root x of $F(x)$ based on a starting point.
- Finds **local optimum** for optimization problem – **initialization strategies** for variables important.

1. Given guess x_k:
   Let $x_{k+1} = x_k + \Delta$
   $F(x_k + \Delta) = F(x_k) + F'(x_k)\Delta + \mathcal{O}(\Delta^2)$
2. Choose update Δ such that
   $F(x_k) + F_x(x_k)\Delta = 0$
3. $x_{k+1} = x_k + \Delta$ and iterate until satisfied



Automatic Differentiation (AD) - compute derivatives efficiently and with ease

- Derivatives are useful and difference approximations are expensive and inaccurate. AD a 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

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

- 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**):

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

Solution of NLP (1/2)

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

- For **equality-constrained NLPs,** i.e., no $h(x)$, 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:

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

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

Solution of NLP (2/2)

- Application of Newton's method for the case with no $h(x)$ gives the iterations as the solution of the **linear equation system**:

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

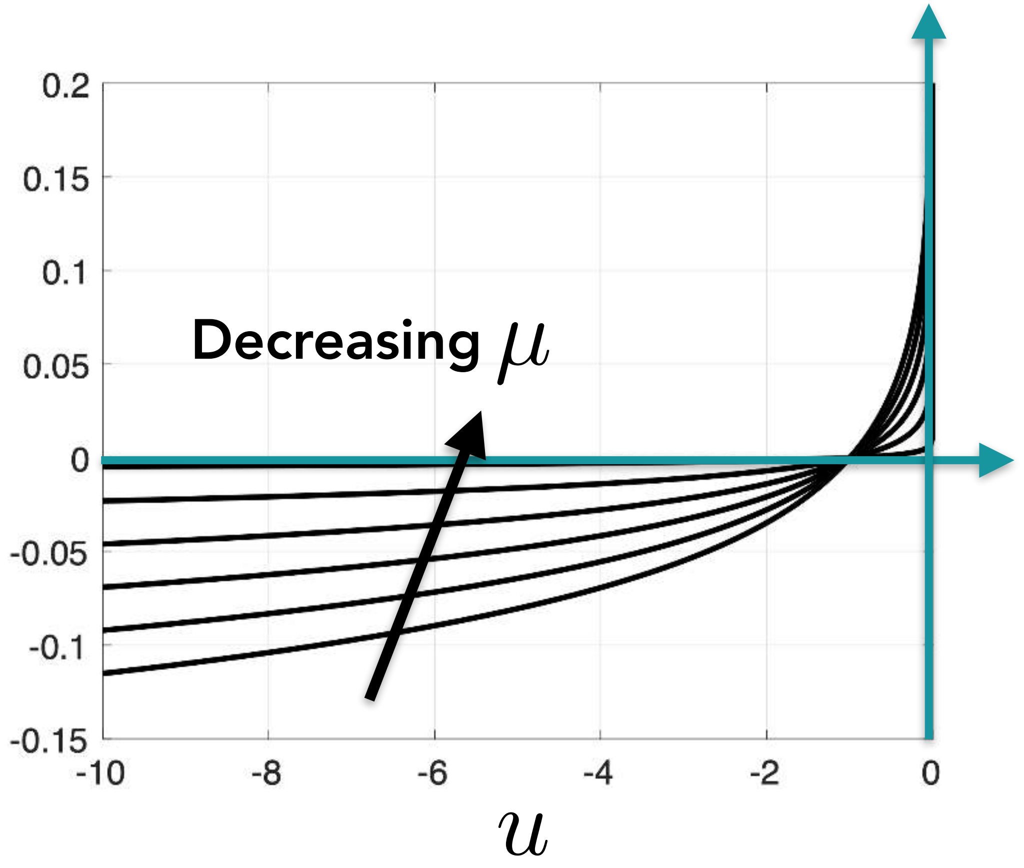
- Requires the **Hessian** of the Lagrangian function.
- Partial Newton step with **line-search** or **trust-region strategies** to ensure intended decrease of specified measure.
- **Quasi-Newton methods** with approximate Hessian exist (of which **BFGS** is one of the most common).

Move inequalities to the loss-function — barrier functions

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

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

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



Interior-Point Methods (1/2)

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

- 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**
- IPOPT is a state-of-the-art implementation of such kind of NLP solver.

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

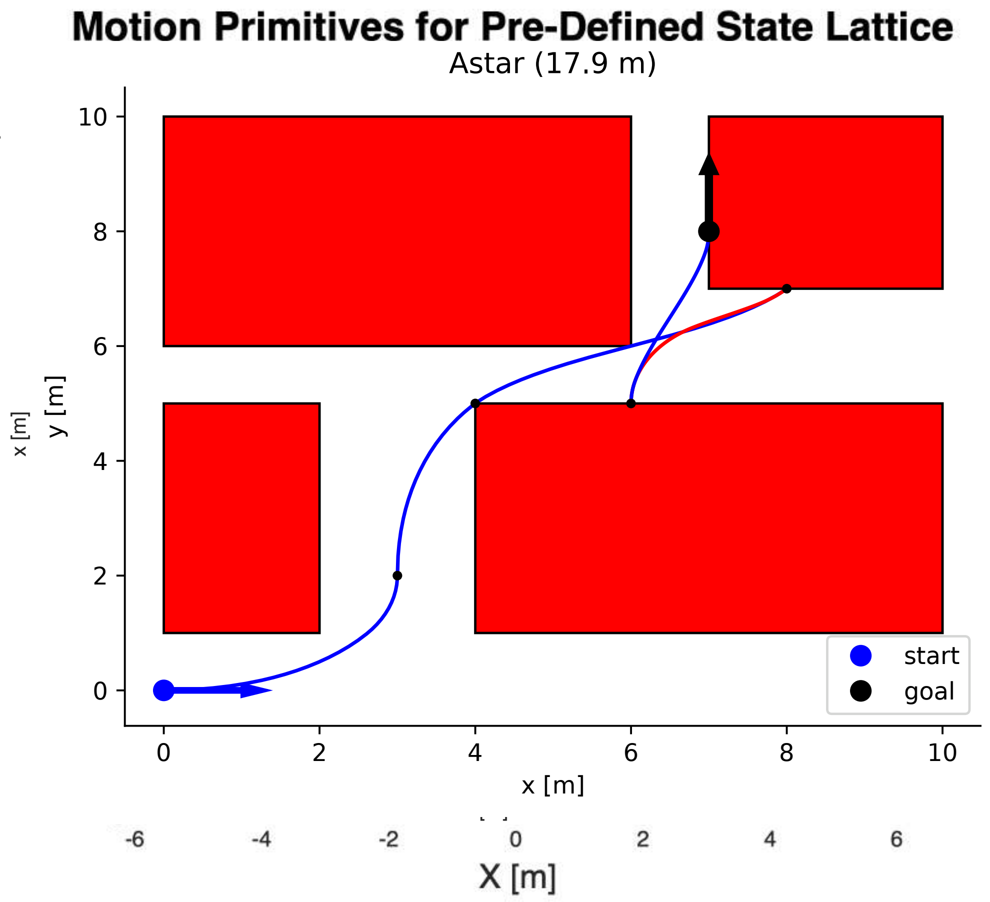
Computing a motion primitive using CasADi and IPOPT

LINKÖPINGS UNIVERSITET

Case Study: Optimization of Motion Primitives (1/6)

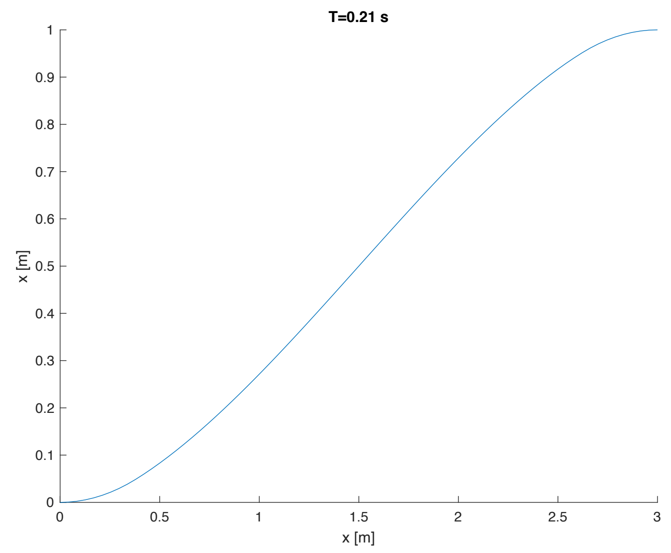
- **Optimization** can be used to compute the **motion primitives** from Lecture 4 (and Hand-in 2)
- Example code using the tool CasADi (py and m-files on course git)
- Vehicle motion equations given by:

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



Case Study: Optimization of Motion Primitives (2/6)

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

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

state_i = [0; 0; 0];  % Initial state (x=0, y=0, theta=0)
state_f = [3; 1; 0];  % Final state (x=3, y=1, theta=0)

opti = casadi.Opti();  % Create optimizer

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

f = casadi.Function('f',{x, u},...
    {v * cos(x(3)), v * sin(x(3)), v * tan(u) / L});
```



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

Case Study: Optimization of Motion Primitives (3/6)

```matlab
X = opti.variable(nx, N + 1);  % Define all state optimization variables
pos_x = X(1, :);
pos_y = X(2, :);
ang_th = X(3, :);

U = opti.variable(N, 1);  % Steer input optimization variables
T = opti.variable(1);  % Final time variable

% Element length
dt = T / N;
```

```matlab
% 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 (4/6)

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

Collocation method: optimize over both x and u simultaneously

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

Details not important here & now ≈ ensure $\dot{x} = f(x, u)$ is satisfied

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

$$-u_{max} \leq u(t) \leq u_{max}$$

Case Study: Optimization of Motion Primitives (5/6)

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

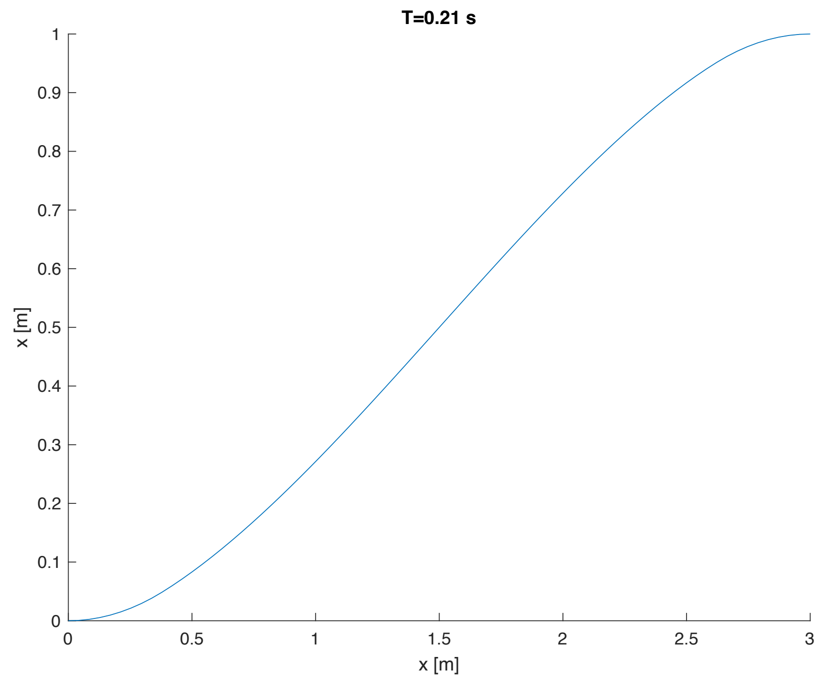
$$x(0) = x_0, x(T) = x_T$$

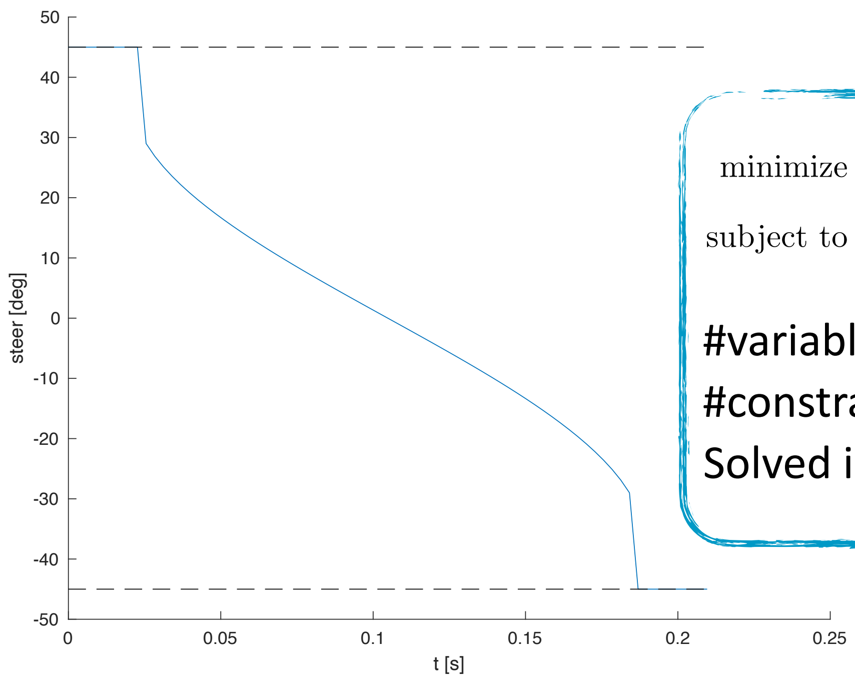
```
% Formulate the cost function
alpha = 1e-2;
opti.minimize(T + alpha*sumsqr(U));
```

minimize $T + \alpha \sum_{k=0}^{N} u(k)^2$

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

Case Study: Optimization of Motion Primitives (6/6)





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

#variables: 979

#constraints: 982

Solved in 80 ms on my machine

References and Further Reading

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.

Take home messages

- General approach for optimal trajectory planning
  - Very useful in Model Predictive Control, see upcoming lecture
- There are algorithms for solving dynamic optimization problems involving motion equations
- Key notions: direct methods, single/multiple shooting, collocation
- For direct methods:
  - Transform dynamic optimization problem to a large non-linear programming (NLP) problem by discretization in different ways.
  - Key step in many solvers — classical Newton-iterationer
  - Efficiency, handling of inequalities — state-of-the-art solvers
- Automatic differentiation — very useful for computing derivatives

www.liu.se

LINKÖPINGS UNIVERSITET