

# AA203

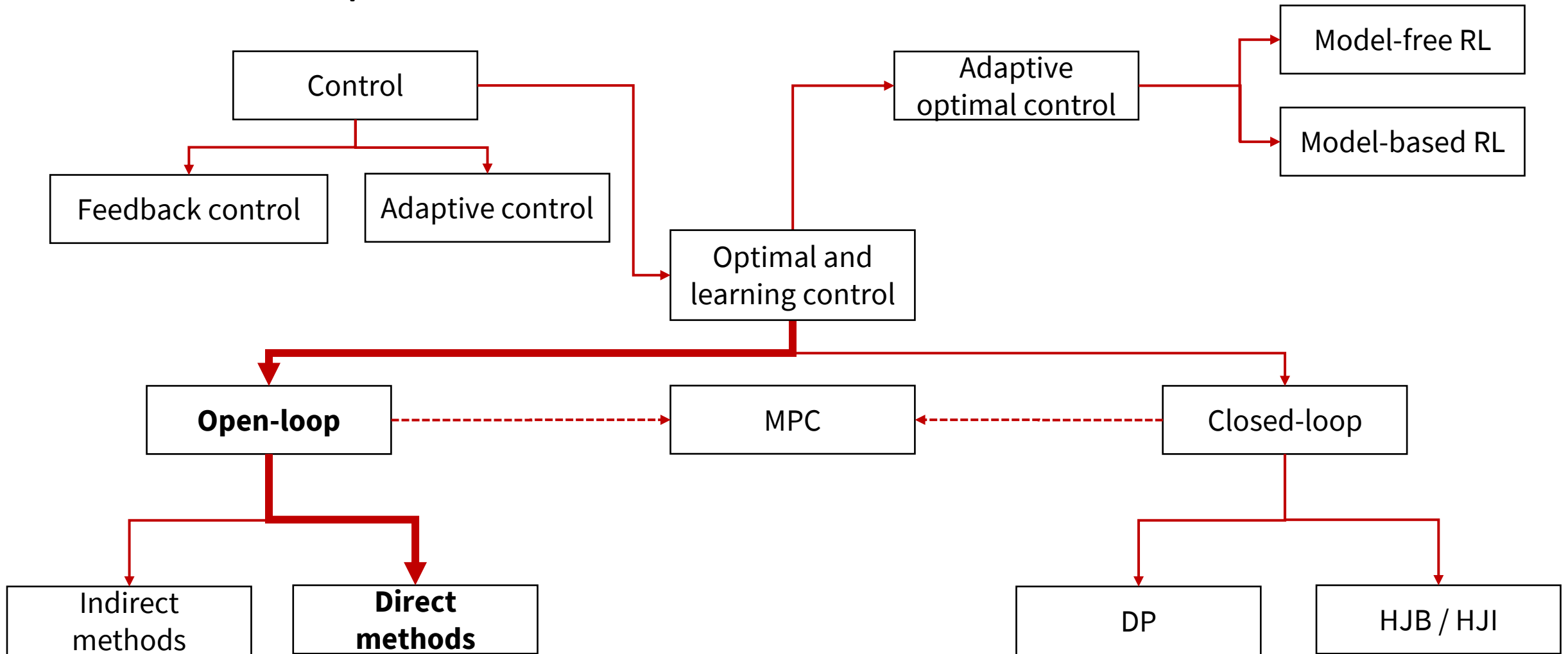
# Optimal and Learning-based Control

Direct methods for optimal control, sequential convex programming (SCP)

# Last time: iLQR and DDP

- Trajectory optimization with a linear feedback tracking policy as a bonus
  - Interpretation as variants of Newton's method in  $Nm$  dimensions
- Drawbacks
  - Output policy applies only locally
  - Dependent on *feasible* initial trajectory
    - (see also [Jur van den Berg, "Extended LQR," 2013.](#))
  - Other than dynamics, only soft-constraints may be incorporated
    - (see also [Howell, et al. "ALTRO," 2019.](#) or [Singh, et al. "Closed-Loop Dynamic SQP," 2021.](#))

# Roadmap



# Optimal control problem

$$\min \int_0^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t) dt$$

**(OCP)**

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t), \quad t \in [0, t_f] \\ \mathbf{x}(0) &= \mathbf{x}_0 \\ \mathbf{x}(t_f) &\in M_f = \{\mathbf{x} \in \mathbb{R}^n : F(\mathbf{x}) = 0\} \\ \mathbf{u}(t) &\in U \subseteq \mathbb{R}^m, \quad t \in [0, t_f]\end{aligned}$$

For simplicity:

- We assume the terminal cost  $h$  is equal to 0
- We assume  $t_0 = 0$

- Direct Methods:
  1. Transcribe **(OCP)** into a nonlinear, constrained optimization problem
  2. Solve the optimization problem via nonlinear programming
- Indirect Methods:
  1. Apply necessary conditions for optimality to **(OCP)**
  2. Solve a two-point boundary value problem

# Direct methods

## Resources:

- [Notes Chapter 5](#) and references therein, and also:
  - [Rao A. V., “A survey of numerical methods for optimal control,” 2009.](#)
  - [Kelly, M., “An Introduction to Trajectory Optimization,” 2017.](#)

# Transcription methods

Optimization: what are the decision variables?

1. State and control parameterization methods
  - “Collocation”/“simultaneous”
2. Control parameterization methods
  - “Shooting”

# Transcription into nonlinear programming (state and control parametrization method)

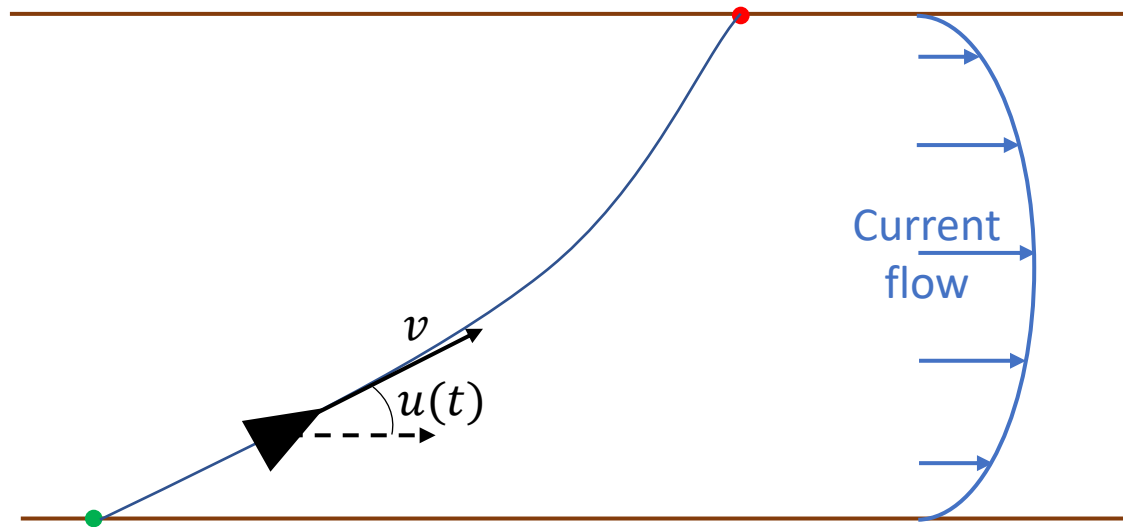
$$\begin{aligned}
 & \min \int_0^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t) dt \\
 & \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t), \quad t \in [0, t_f] \\
 & \text{(OCP)} \quad \mathbf{x}(0) = \mathbf{x}_0 \\
 & \quad \mathbf{x}(t_f) \in M_f = \{\mathbf{x} \in \mathbb{R}^n : F(\mathbf{x}) = 0\} \\
 & \quad \mathbf{u}(t) \in U \subseteq \mathbb{R}^m, \quad t \in [0, t_f]
 \end{aligned}$$

$$\begin{aligned}
 & \min_{(\mathbf{x}_i, \mathbf{u}_i)} \sum_{i=0}^{N-1} h_i g(\mathbf{x}_i, \mathbf{u}_i, t_i) \\
 & \text{(NLOP)} \quad \mathbf{x}_{i+1} = \mathbf{x}_i + h_i \mathbf{f}(\mathbf{x}_i, \mathbf{u}_i, t_i), \quad i = 0, \dots, N-1 \\
 & \quad \mathbf{u}_i \in U, \quad i = 0, \dots, N-1, \quad F(\mathbf{x}_N) = 0
 \end{aligned}$$

## Forward Euler time discretization

1. Select a discretization  $0 = t_0 < t_1 < \dots < t_N = t_f$  for the interval  $[0, t_f]$  and, for every  $i = 0, \dots, N-1$ , define  $\mathbf{x}_i \sim \mathbf{x}(t)$ ,  $\mathbf{u}_i \sim \mathbf{u}(t)$ ,  $t \in [t_i, t_{i+1})$  and  $\mathbf{x}_0 \sim \mathbf{x}(0)$
2. By denoting  $h_i = t_{i+1} - t_i$ , (OCP) is transcribed into the following nonlinear, constrained optimization problem

# Illustrative example: Zermelo's Problem



$$\begin{aligned} & \min \int_0^{t_f} u(t)^2 dt \\ & \dot{x}(t) = v \cos(u(t)) + \text{flow}(y(t)), t \in [0, t_f] \\ \text{(OCP)} \quad & \dot{y}(t) = v \sin(u(t)), t \in [0, t_f] \\ & (x, y)(0) = 0, (x, y)(t_f) = (M, \ell) \\ & |u(t)| \leq u_{\max}, t \in [0, t_f] \end{aligned}$$



# Example: Zermelo's Problem

## State and control parameterization method

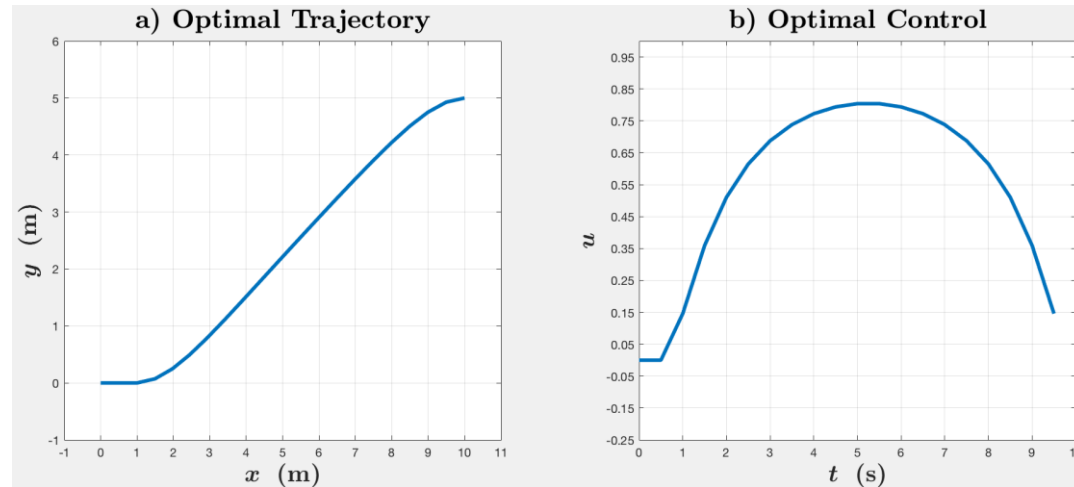
- Transcribe optimal control problem into a non-linear program, and solve it via `fmincon` (MATLAB), `scipy.optimize.minimize` (python), etc.

$$\begin{aligned} \min \quad & \int_0^{t_f} u(t)^2 dt \\ \text{(OCP)} \quad & \dot{x}(t) = v \cos(u(t)) + \text{flow}(y(t)), t \in [0, t_f] \\ & \dot{y}(t) = v \sin(u(t)), t \in [0, t_f] \\ & (x, y)(0) = 0, (x, y)(t_f) = (M, \ell) \\ & |u(t)| \leq u_{\max}, t \in [0, t_f] \end{aligned}$$

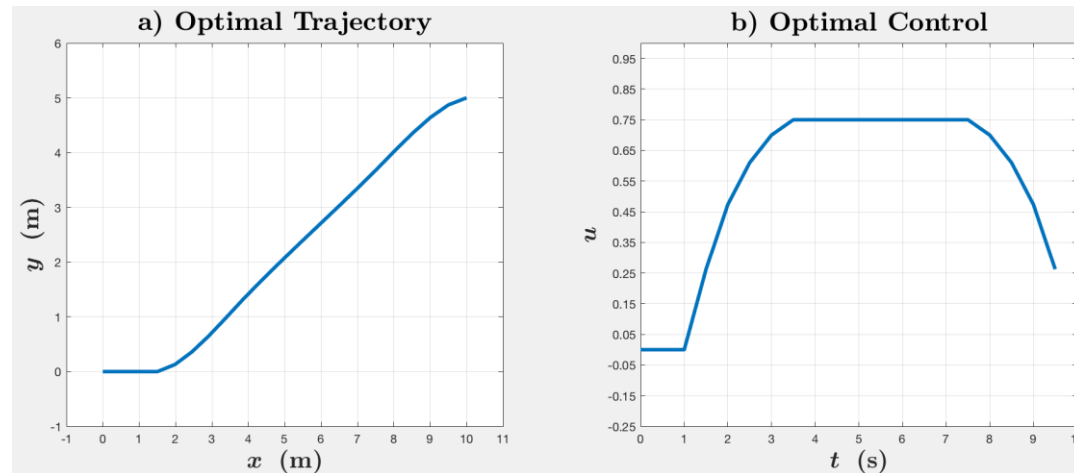


$$\begin{aligned} \min_{(x_i, u_i)} \quad & \sum_{i=0}^{N-1} h u_i^2 \\ \text{(NLOP)} \quad & x_{i+1} = x_i + h(v \cos(u_i) + \text{flow}(y_i)) \\ & y_{i+1} = y_i + h v \sin(u_i), |u_i| \leq u_{\max} \\ & (x_0, y_0) = 0, (x_N, y_N) = (M, \ell) \end{aligned}$$

# Results



$|u(t)| \leq 1$   
(effectively, no control  
constraint)



$|u(t)| \leq 0.75$

# Transcription into nonlinear programming (control parametrization method)

$$\begin{aligned} & \min \int_0^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t) dt \\ & \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t), \quad t \in [0, t_f] \\ \text{(OCP)} \quad & \mathbf{x}(0) = \mathbf{x}_0 \\ & \mathbf{x}(t_f) \in M_f = \{\mathbf{x} \in \mathbb{R}^n : F(\mathbf{x}) = 0\} \\ & \mathbf{u}(t) \in U \subseteq \mathbb{R}^m, \quad t \in [0, t_f] \end{aligned}$$

$$\begin{aligned} & \min_{\mathbf{u}_i} \sum_{i=0}^{N-1} h_i g(\mathbf{x}(t_i), \mathbf{u}_i, t_i) \\ \text{(NLOP-C)} \quad & \mathbf{u}_i \in U, \quad i = 0, \dots, N-1, \quad F(\mathbf{x}(t_N)) = 0 \end{aligned}$$

where each  $\mathbf{x}(t_i)$  is recursively computed via  
 $\mathbf{x}(t_{i+1}) = \mathbf{x}(t_i) + h_i \mathbf{f}(\mathbf{x}(t_i), \mathbf{u}_i, t_i), \quad i = 0, \dots, N-1$

## Time and control discretization

1. Select a discretization  $0 = t_0 < t_1 < \dots < t_N = t_f$  for the interval  $[0, t_f]$  and, for every  $i = 0, \dots, N-1$ , define  
 $\mathbf{u}_i \sim \mathbf{u}(t), \quad t \in [t_i, t_{i+1})$
2. By denoting  $h_i = t_{i+1} - t_i$ , (OCP) is transcribed into the following nonlinear, constrained optimization problem

# Example: Zermelo's Problem

## Control parameterization method

- Transcribe optimal control problem into a non-linear program, and solve it via `fmincon` (MATLAB), `scipy.optimize.minimize` (python), etc.

$$\begin{aligned} & \min \int_0^{t_f} u(t)^2 dt \\ & \dot{x}(t) = v \cos(u(t)) + \text{flow}(y(t)), t \in [0, t_f] \\ \text{(OCP)} \quad & \dot{y}(t) = v \sin(u(t)), t \in [0, t_f] \\ & (x, y)(0) = 0, (x, y)(t_f) = (M, \ell) \\ & |u(t)| \leq u_{\max}, t \in [0, t_f] \end{aligned}$$



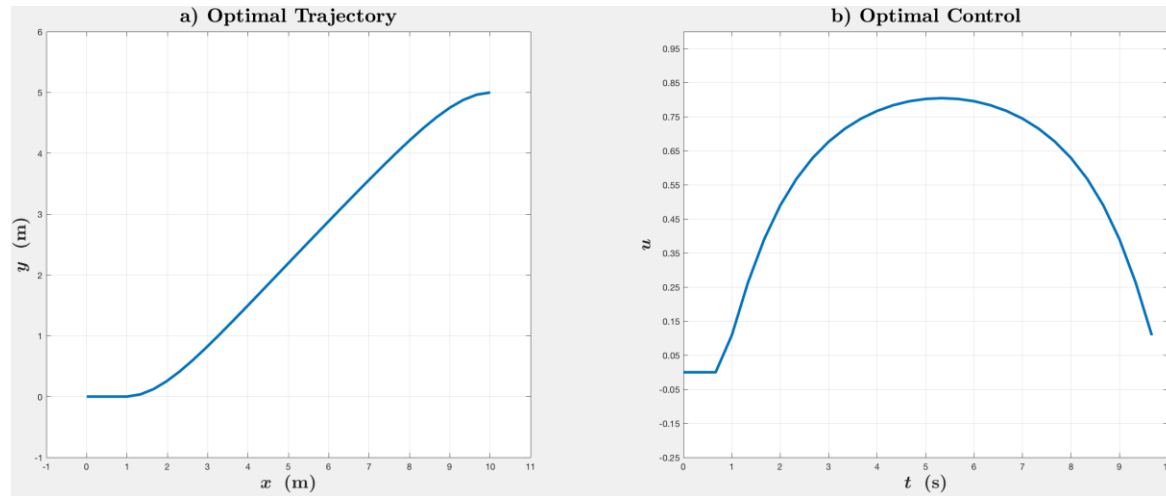
$$\min_{u_i} \sum_{i=0}^{N-1} h u_i^2 \quad \text{(NLOP-C)}$$

$$(x, y)(t_N) = (M, \ell), \quad |u_i| \leq u_{\max}$$

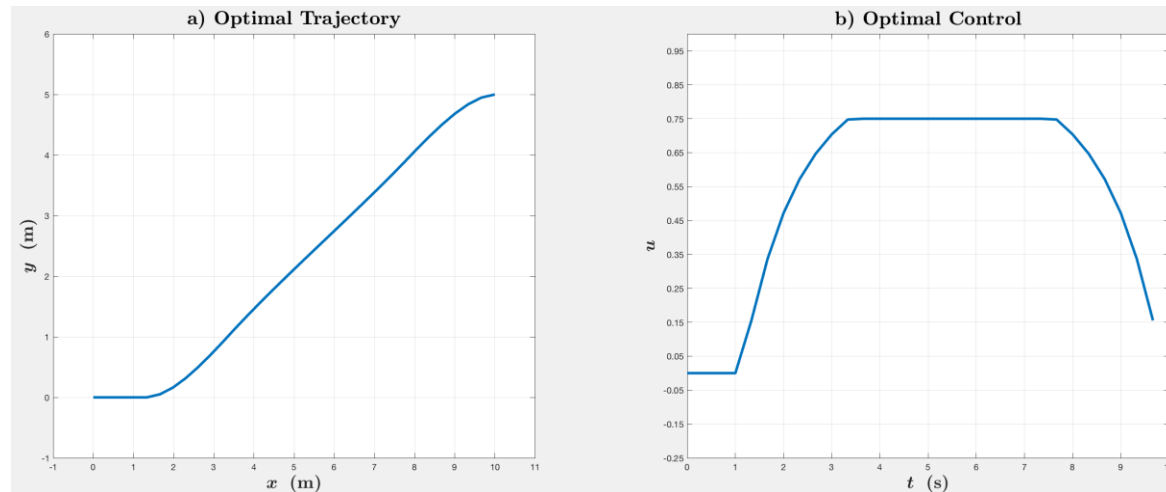
where, recursively:

$$x_N = x_0 + h \sum_{i=0}^{N-1} (v \cos(u_i) + \text{flow}(y_i)), \quad y_i = y_0 + h \sum_{j=0}^i v \sin(u_j)$$

# Results



$|u(t)| \leq 1$   
(effectively, no control  
constraint)



$|u(t)| \leq 0.75$

# Example: Zermelo's Problem

**(OCP)**

$$\min \int_0^{t_f} u(t)^2 dt$$

$$\dot{x}(t) = v \cos(u(t)) + \text{flow}(y(t)), t \in [0, t_f]$$

$$\dot{y}(t) = v \sin(u(t)), t \in [0, t_f]$$

$$(x, y)(0) = 0, (x, y)(t_f) = (M, \ell)$$

$$|u(t)| \leq u_{\max}, t \in [0, t_f]$$



**(NLOP)**

$$\min_{(x_i, u_i)} \sum_{i=0}^{N-1} h u_i^2$$

$$x_{i+1} = x_i + h(v \cos(u_i) + \text{flow}(y_i))$$

$$y_{i+1} = y_i + h v \sin(u_i), |u_i| \leq u_{\max}$$

$$(x_0, y_0) = 0, (x_N, y_N) = (M, \ell)$$

Direct Transcription

**(NLOP-C)**

$$\min_{u_i} \sum_{i=0}^{N-1} h u_i^2$$

$$(x, y)(t_N) = (M, \ell), \quad |u_i| \leq u_{\max}$$

where, recursively:

$$x_N = x_0 + h \sum_{i=0}^{N-1} (v \cos(u_i) + \text{flow}(y_i))$$

$$y_i = y_0 + h \sum_{j=0}^i v \sin(u_j)$$

Direct Shooting

# Transcription methods: extensions

- Multiple shooting
  - Hybrid of simultaneous / (single) shooting methods
- Alternative trajectory parameterizations
  - Euler integration (above): piecewise linear effective state trajectory ( $C^0$ ), zero-order hold control trajectory
  - Hermite-Simpson collocation (see [Notes §5.2.1](#)): piecewise cubic effective state trajectory ( $C^1$ ), first-order hold control trajectory
    - Dynamics constraint is enforced at “collocation points,” exact form is derived by implicit integration
  - Pseudospectral methods: global polynomial basis functions (instead of piecewise polynomials)
  - Shooting methods: higher-order integration schemes (e.g., [RK4](#))
    - Dynamics constraint is enforced by explicit integration

# Sequential Convex Programming

$$\begin{aligned} & \min \int_0^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t) dt \\ (\mathbf{OCP}) \quad & \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t), \quad t \in [0, t_f] \\ & \mathbf{x}(0) = \mathbf{x}_0, \quad \mathbf{x}(t_f) = \mathbf{x}_f \\ & \mathbf{u}(t) \in U \subseteq \mathbb{R}^m, \quad t \in [0, t_f] \end{aligned}$$

The sources of nonconvexities are the dynamics and (possibly) the cost. **Idea: linearize (and convexify) them around nominal trajectories!**



# Sequential Convex Programming

$$\begin{aligned} & \min \int_0^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t) dt \\ (\text{OCP}) \quad & \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t), \quad t \in [0, t_f] \\ & \mathbf{x}(0) = \mathbf{x}_0, \quad \mathbf{x}(t_f) = \mathbf{x}_f \\ & \mathbf{u}(t) \in U \subseteq \mathbb{R}^m, \quad t \in [0, t_f] \end{aligned}$$

The sources of nonconvexities are the dynamics and (possibly) the cost. **Idea: linearize (and convexify) them around nominal trajectories!**

1. Assume that  $g$  is convex. Let  $(\mathbf{x}_0(\cdot), \mathbf{u}_0(\cdot))$  be a nominal tuple of trajectory and control.  **$(\mathbf{x}_0(\cdot), \mathbf{u}_0(\cdot))$  does not need to be feasible!**

# Sequential Convex Programming

$$\begin{aligned} & \min \int_0^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t) dt \\ (\text{OCP}) \quad & \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t), \quad t \in [0, t_f] \\ & \mathbf{x}(0) = \mathbf{x}_0, \quad \mathbf{x}(t_f) = \mathbf{x}_f \\ & \mathbf{u}(t) \in U \subseteq \mathbb{R}^m, \quad t \in [0, t_f] \end{aligned}$$

The sources of nonconvexities are the dynamics and (possibly) the cost. **Idea: linearize (and convexify) them around nominal trajectories!**

1. Assume that  $g$  is convex. Let  $(\mathbf{x}_0(\cdot), \mathbf{u}_0(\cdot))$  be a nominal tuple of trajectory and control.  **$(\mathbf{x}_0(\cdot), \mathbf{u}_0(\cdot))$  does not need to be feasible!**
2. Linearize  $\mathbf{f}$  around  $(\mathbf{x}_0(\cdot), \mathbf{u}_0(\cdot))$ :

$$\begin{aligned} & \mathbf{f}_1(\mathbf{x}, \mathbf{u}, t) \\ &= \mathbf{f}(\mathbf{x}_0(t), \mathbf{u}_0(t), t) + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}_0(t), \mathbf{u}_0(t), t)(\mathbf{x} - \mathbf{x}_0(t)) + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{x}_0(t), \mathbf{u}_0(t), t)(\mathbf{u} - \mathbf{u}_0(t)) \end{aligned}$$

# Sequential Convex Programming

$$\begin{aligned} & \min \int_0^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t) dt \\ (\text{LOCP})_1 \quad & \dot{\mathbf{x}}(t) = \mathbf{f}_1(\mathbf{x}(t), \mathbf{u}(t), t), \quad t \in [0, t_f] \\ & \mathbf{x}(0) = \mathbf{x}_0, \quad \mathbf{x}(t_f) = \mathbf{x}_f \\ & \mathbf{u}(t) \in U \subseteq \mathbb{R}^m, \quad t \in [0, t_f] \end{aligned}$$

The sources of nonconvexities are the dynamics and (possibly) the cost. **Idea: linearize (and convexify) them around nominal trajectories!**

1. Assume that  $g$  is convex. Let  $(\mathbf{x}_0(\cdot), \mathbf{u}_0(\cdot))$  be a nominal tuple of trajectory and control.  **$(\mathbf{x}_0(\cdot), \mathbf{u}_0(\cdot))$  does not need to be feasible!**
2. Linearize  $\mathbf{f}$  around  $(\mathbf{x}_0(\cdot), \mathbf{u}_0(\cdot))$ :  
$$\begin{aligned} & \mathbf{f}_1(\mathbf{x}, \mathbf{u}, t) \\ &= \mathbf{f}(\mathbf{x}_0(t), \mathbf{u}_0(t), t) + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}_0(t), \mathbf{u}_0(t), t)(\mathbf{x} - \mathbf{x}_0(t)) + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{x}_0(t), \mathbf{u}_0(t), t)(\mathbf{u} - \mathbf{u}_0(t)) \end{aligned}$$
3. Solve the new **problem  $(\text{LOCP})_1$**  for  $(\mathbf{x}_1(\cdot), \mathbf{u}_1(\cdot))$

# Sequential Convex Programming

$$\begin{aligned} & \min \int_0^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t) dt \\ & \dot{\mathbf{x}}(t) = \mathbf{f}_{k+1}(\mathbf{x}(t), \mathbf{u}(t), t), \quad t \in [0, t_f] \\ (\text{LOCP})_{k+1} \quad & \mathbf{x}(0) = \mathbf{x}_0, \quad \mathbf{x}(t_f) = \mathbf{x}_f \\ & \mathbf{u}(t) \in U \subseteq \mathbb{R}^m, \quad t \in [0, t_f] \end{aligned}$$

The sources of nonconvexities are the dynamics and (possibly) the cost. **Idea: linearize (and convexify) them around nominal trajectories!**

4. Iterate this procedure until convergence is achieved: linearize  $\mathbf{f}$  around the solution  $(\mathbf{x}_k(\cdot), \mathbf{u}_k(\cdot))$  at iteration  $k$ :

$$\begin{aligned} & \mathbf{f}_{k+1}(\mathbf{x}, \mathbf{u}, t) \\ &= \mathbf{f}(\mathbf{x}_k(t), \mathbf{u}_k(t), t) + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}_k(t), \mathbf{u}_k(t), t)(\mathbf{x} - \mathbf{x}_k(t)) + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{x}_k(t), \mathbf{u}_k(t), t)(\mathbf{u} - \mathbf{u}_k(t)) \\ & \text{and solve the problem } (\text{LOCP})_{k+1} \text{ for } (\mathbf{x}_{k+1}(\cdot), \mathbf{u}_{k+1}(\cdot)) \end{aligned}$$

# Sequential Convex Programming

$$\begin{aligned}
 (\mathbf{LOCP})_{k+1} \quad & \min \int_0^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t) \, dt \\
 & \dot{\mathbf{x}}(t) = \mathbf{f}_{k+1}(\mathbf{x}(t), \mathbf{u}(t), t), \quad t \in [0, t_f] \\
 & \mathbf{x}(0) = \mathbf{x}_0, \quad \mathbf{x}(t_f) = \mathbf{x}_f \\
 & \mathbf{u}(t) \in U \subseteq \mathbb{R}^m, \quad t \in [0, t_f]
 \end{aligned}$$

Discretize and Solve a Convex Problem at Each Iteration

1. Select a discretization  $0 = t_0 < t_1 < \dots < t_N = t_f$  for the interval  $[0, t_f]$  and, for every  $i = 0, \dots, N - 1$ , define  $\mathbf{x}_{i+1} \sim \mathbf{x}(t)$ ,  $\mathbf{u}_i \sim \mathbf{u}(t)$ ,  $t \in (t_i, t_{i+1}]$  and  $\mathbf{x}_0 \sim \mathbf{x}(0)$
2. By denoting  $h_i = t_{i+1} - t_i$ ,  $(\mathbf{LOCP})_{k+1}$  is transcribed into the following **convex optimization problem**

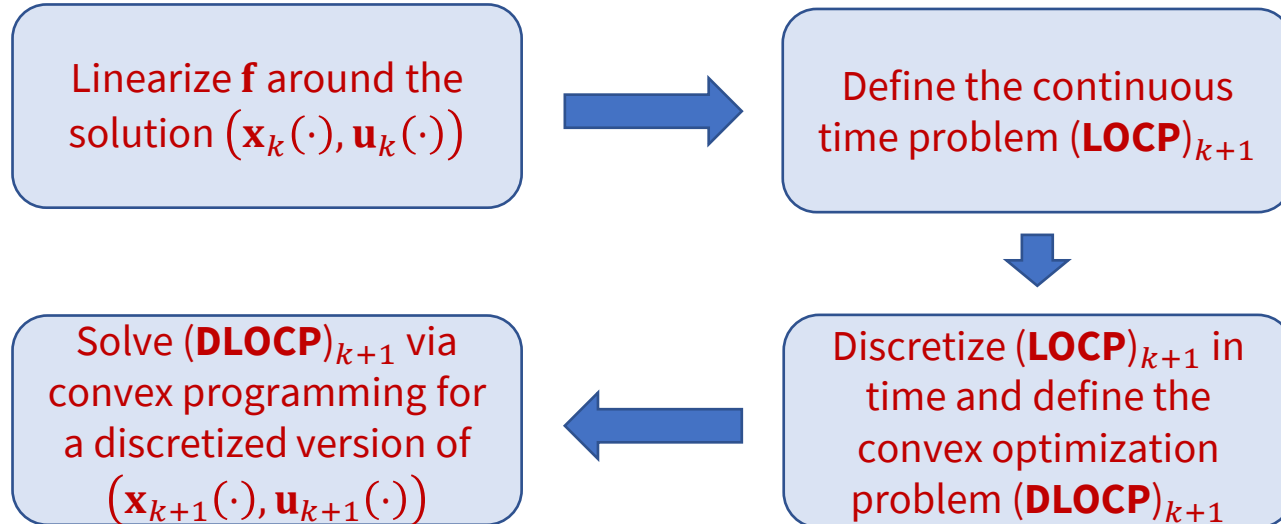
$$\begin{aligned}
 (\mathbf{DLOCP})_{k+1} \quad & \min_{(\mathbf{x}_i, \mathbf{u}_i)} \sum_{i=0}^{N-1} h_i g(\mathbf{x}_i, \mathbf{u}_i, t_i) \\
 & \mathbf{x}_{i+1} = \mathbf{x}_i + h_i \mathbf{f}_{k+1}(\mathbf{x}_i, \mathbf{u}_i, t_i), \quad i = 0, \dots, N - 1 \\
 & \mathbf{u}_i \in U, \quad i = 0, \dots, N - 1, \quad \mathbf{x}_N = \mathbf{x}_f
 \end{aligned}$$

# Sequential Convex Programming

$$\begin{aligned}
 & \min \int_0^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t) dt \\
 & \dot{\mathbf{x}}(t) = \mathbf{f}_{k+1}(\mathbf{x}(t), \mathbf{u}(t), t), \quad t \in [0, t_f] \\
 (\mathbf{LOCP})_{k+1} \quad & \mathbf{x}(0) = \mathbf{x}_0, \quad \mathbf{x}(t_f) = \mathbf{x}_f \\
 & \mathbf{u}(t) \in U \subseteq \mathbb{R}^m, \quad t \in [0, t_f]
 \end{aligned}$$

$$\begin{aligned}
 & \min_{(\mathbf{x}_i, \mathbf{u}_i)} \sum_{i=0}^{N-1} h_i g(\mathbf{x}_i, \mathbf{u}_i, t_i) \\
 (\mathbf{DLOCP})_{k+1} \quad & \mathbf{x}_{i+1} = \mathbf{x}_i + h_i \mathbf{f}_{k+1}(\mathbf{x}_i, \mathbf{u}_i, t_i), \quad i = 0, \dots, N-1 \\
 & \mathbf{u}_i \in U, \quad i = 0, \dots, N-1, \quad \mathbf{x}_N = \mathbf{x}_f
 \end{aligned}$$

SCP Methodology: at each iteration  $k$ ,



# Direct Methods in Practice

“As you begin to play with these algorithms on your own problems, you might feel like you're on an emotional roller-coaster.” – [Russ Tedrake](#)

- Better initial guess trajectories (“warm-starting” the optimization, as seen in `zermelo_simultaneous`)
- Cost function/constraint tuning (as seen in `zermelo_scp`)
  - Penalty methods; augmented Lagrangian-based solvers

# Next time

- Dynamic programming  
in continuous time