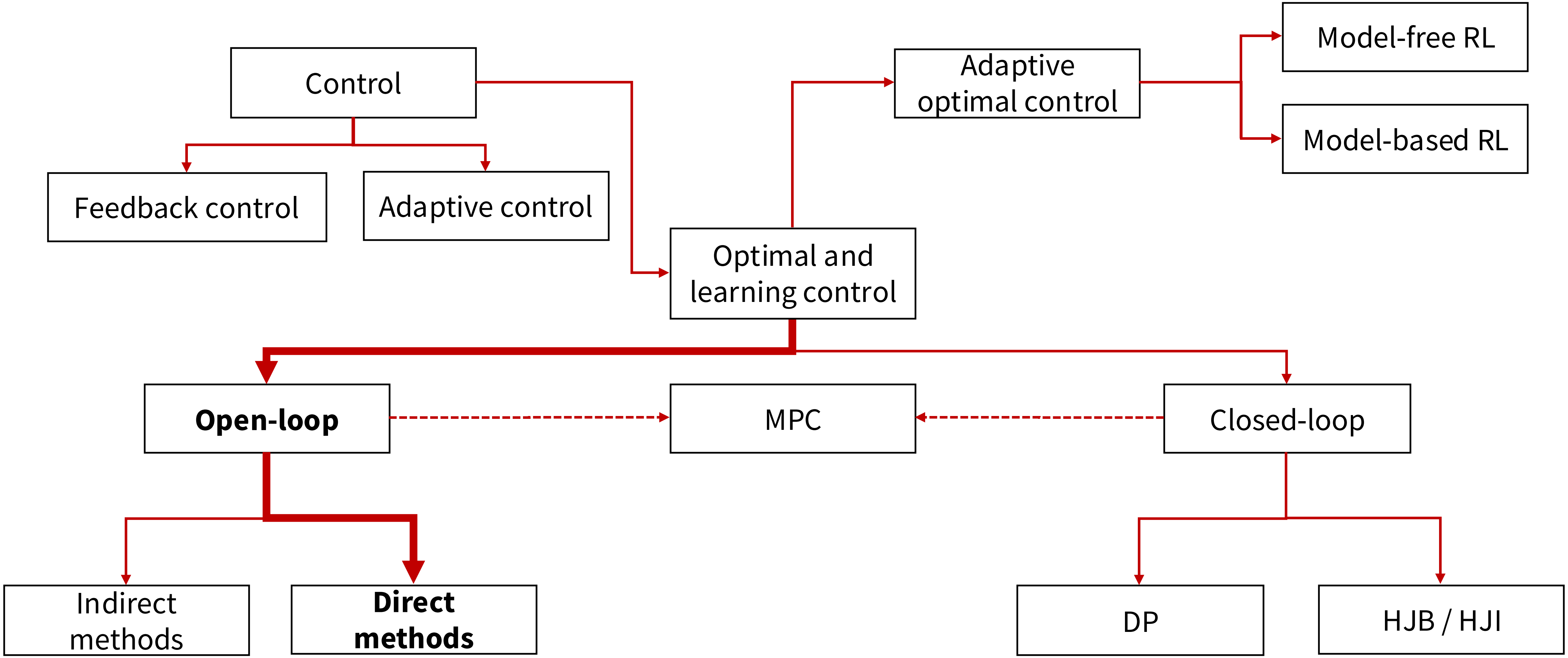
AA203
Optimal and Learning-based Control

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

Stanford University

Autonomous Systems Lab

Roadmap


Control
Feedback control
Adaptive control
Optimal and learning control
Adaptive optimal control
Model-free RL
Model-based RL
Open-loop
MPC
Closed-loop
Indirect methods
Direct methods
DP
HJB / HJI

Optimal control problem

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

(OCP)

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t), \;\; t \in [0, t_f]$$

$$\mathbf{x}(0) = \mathbf{x}_0$$

$$\mathbf{x}(t_f) \in M_f = \{\mathbf{x} \in \mathbb{R}^n : F(\mathbf{x}) = 0\}$$

$$\mathbf{u}(t) \in U \subseteq \mathbb{R}^m, \quad t \in [0, t_f]$$

For simplicity:

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

- Indirect Methods:
  1. Apply necessary conditions for optimality to (**OCP**)
  2. Solve a two-point boundary value problem

```python
def free_final_time_2pbvp(a=1.0, b=1.0, N=20):
    # Indirect method (solving a two-point boundary value problem).

    def ode(t, x_p_tf):
        x1, x2, p1, p2, tf = x_p_tf
        return tf * np.array([x2, -p2 / b, np.zeros_like(t), -p1, np.zeros_like(t)])

    def boundary_conditions(x_p_tf_0, x_p_tf_N):
        x1_0, x2_0, p1_0, p2_0, tf_0 = x_p_tf_0
        x1_N, x2_N, p1_N, p2_N, tf_N = x_p_tf_N
        return np.array([x1_0 - 10, x2_0, x1_N, x2_N, a * tf_N - p2_N**2 / (2 * b)])

    return solve_bvp(
        fun=ode, bc=boundary_conditions, x=np.linspace(0, 1, N + 1),
        y=np.array([np.linspace(10, 0, N + 1), np.zeros(N + 1), np.zeros(N + 1), np.zeros(N + 1), np.ones(N + 1)]))
```

- Direct Methods:
  1. Transcribe (**OCP**) into a nonlinear, constrained optimization problem
  2. Solve the optimization problem via nonlinear programming

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)

(OCP)

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

$\Rightarrow$

(NLOP)

$$\min_{(\mathbf{x}_i, \mathbf{u}_i)} \sum_{i=0}^{N-1} h_i g(\mathbf{x}_i, \mathbf{u}_i, t_i)$$
$$\mathbf{x}_{i+1} = \mathbf{x}_i + h_i \mathbf{f}(\mathbf{x}_i, \mathbf{u}_i, t_i), \qquad i = 0, \ldots, N-1$$
$$\mathbf{u}_i \in U,\ i = 0, \ldots, N-1, \qquad F(\mathbf{x}_N) = 0$$

Forward Euler time discretization:

1. Select a discretization $0 = t_0 < t_1 < \cdots < t_N = t_f$ for the interval $[0, t_f]$ and, for every $i = 0, \ldots, N-1$, define $\mathbf{x}_i \sim \mathbf{x}(t)$, $\mathbf{u}_i \sim \mathbf{u}(t)$, $t \in [t_i, t_{i+1})$ and $\mathbf{x}_0 \sim \mathbf{x}(0)$

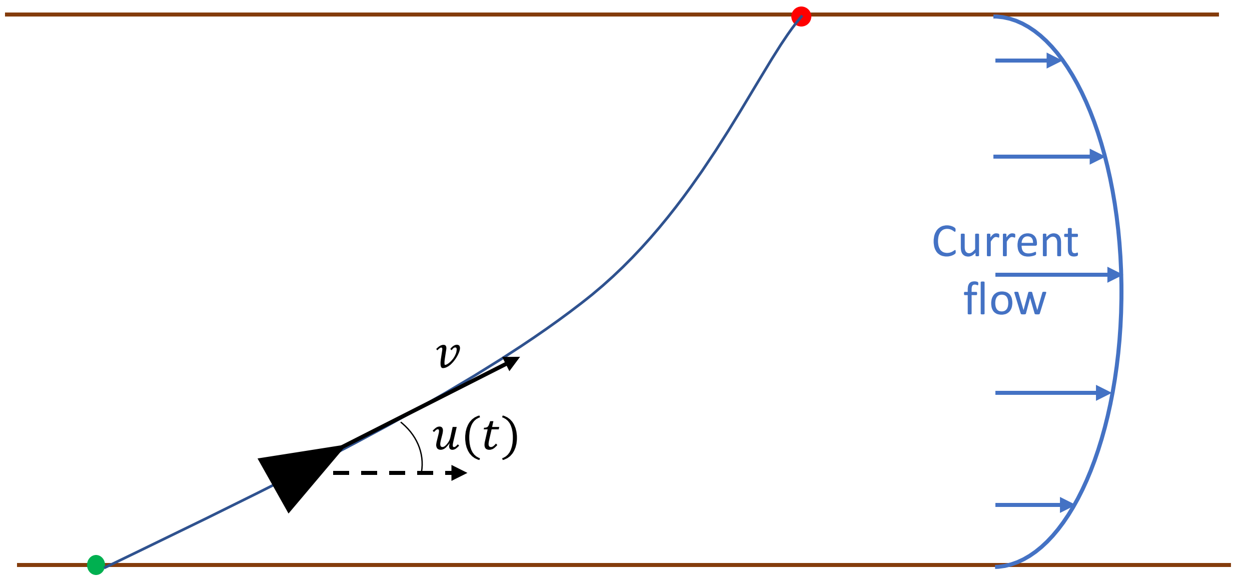
2. By denoting $h_i = t_{i+1} - t_i$, (**OCP**) is transcribed into a nonlinear, constrained optimization problem

Illustrative example: Zermelo's Problem

(OCP)

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

$$\dot{x}(t) = v\cos\big(u(t)\big) + \text{flow}\big(y(t)\big),\ t \in [0, t_f]$$
$$\dot{y}(t) = v\sin\big(u(t)\big),\ t \in [0, t_f]$$
$$(x,y)(0) = 0,\ (x,y)(t_f) = (M,\ell)$$
$$|u(t)| \le u_{max},\ t \in [0, t_f]$$



Example: Zermelo's Problem
(state and control parametrization method)

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

(OCP)

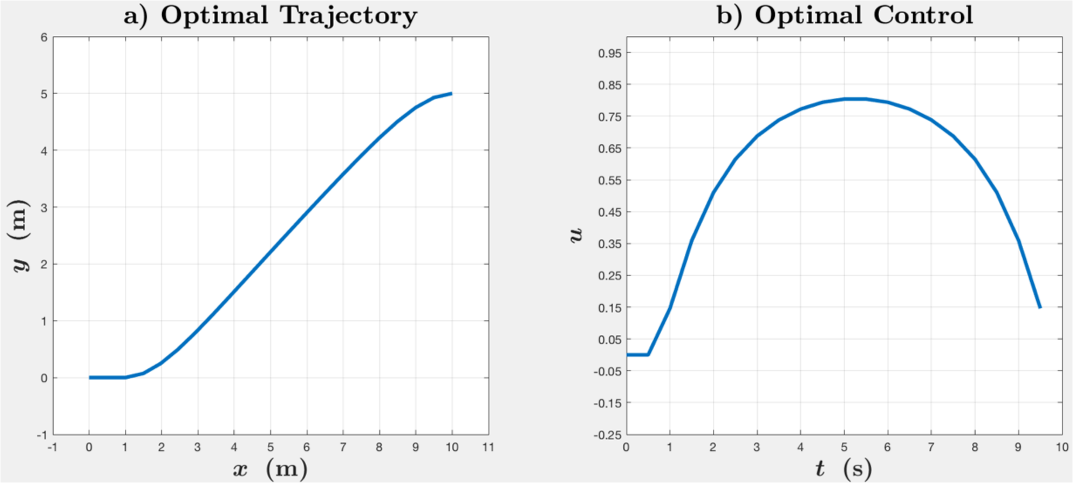
$$\min \int_0^{t_f} u(t)^2 \, dt$$
$$\dot{x}(t) = v\cos\big(u(t)\big) + \mathrm{flow}\big(y(t)\big),\ t \in [0, t_f]$$
$$\dot{y}(t) = v\sin\big(u(t)\big),\ t \in [0, t_f]$$
$$(x, y)(0) = 0,\ (x, y)(t_f) = (M, \ell)$$
$$|u(t)| \leq u_{max},\ t \in [0, t_f]$$

$\Rightarrow$

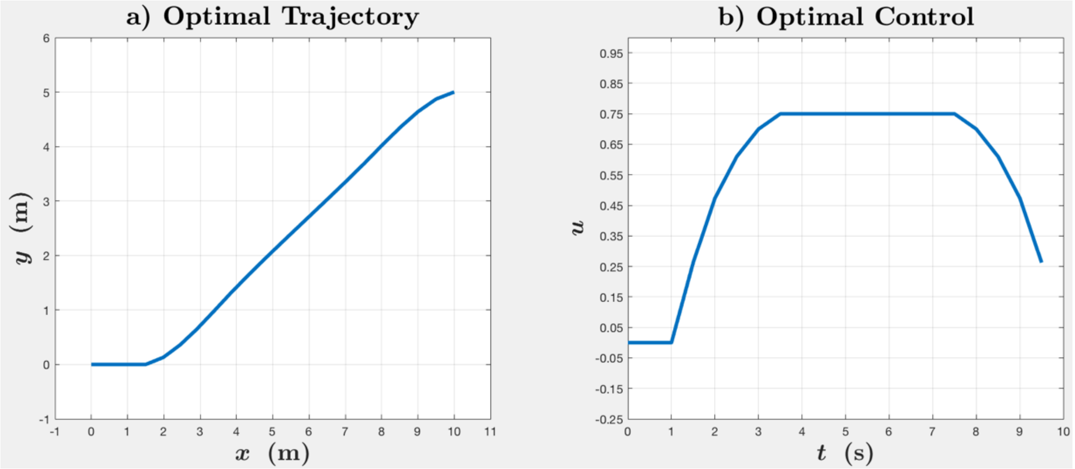
(NLOP)

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

Results



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



$|u(t)| \leq 0.75$

Transcription into nonlinear programming
(control parametrization method)

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

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

(**NLOP-C**) $$\min_{\mathbf{u}_i} \sum_{i=0}^{N-1} h_i g(\mathbf{x}(t_i), \mathbf{u}_i, t_i)$$

$$\mathbf{u}_i \in U\ , i = 0, \dots, N-1\ , \qquad F(\mathbf{x}(t_N)) = 0$$

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

Time and control discretization:

1. Select a discretization $0 = t_0 < t_1 < \cdots < t_N = t_f$ for the interval $[0, t_f]$ and, for every $i = 0, \dots, N-1$, define $\mathbf{u}_i \sim \mathbf{u}(t)\ ,\ \ t \in [t_i, t_{i+1})$

2. By denoting $h_i = t_{i+1} - t_i$, (**OCP**) is transcribed into a nonlinear, constrained optimization problem

Example: Zermelo's Problem

(control parametrization method)

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

(OCP)

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

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

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

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

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

(NLOP-C)

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

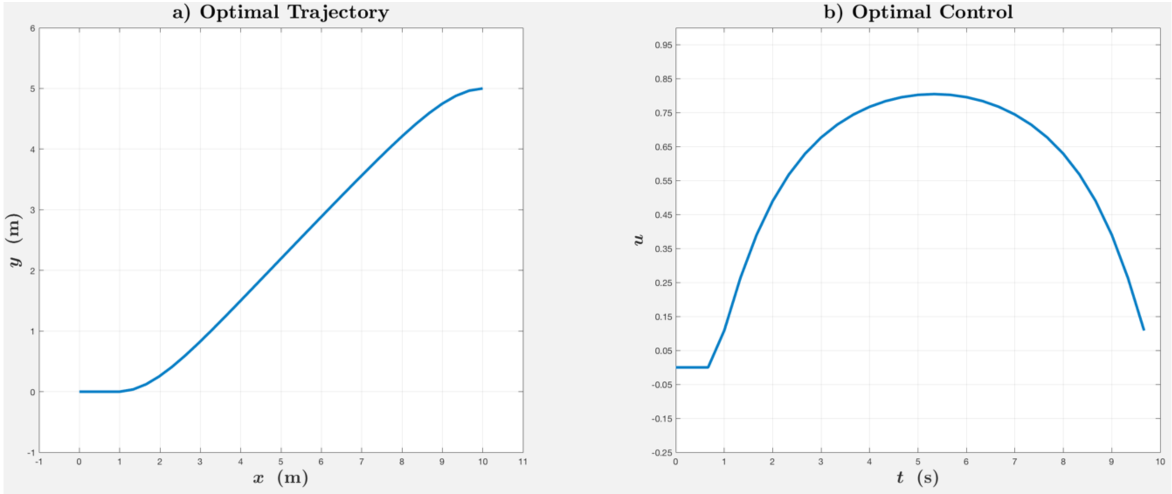
$$(x, y)(t_N) = (M, \ell), \quad |u_i| \le u_{max}$$

where, recursively:

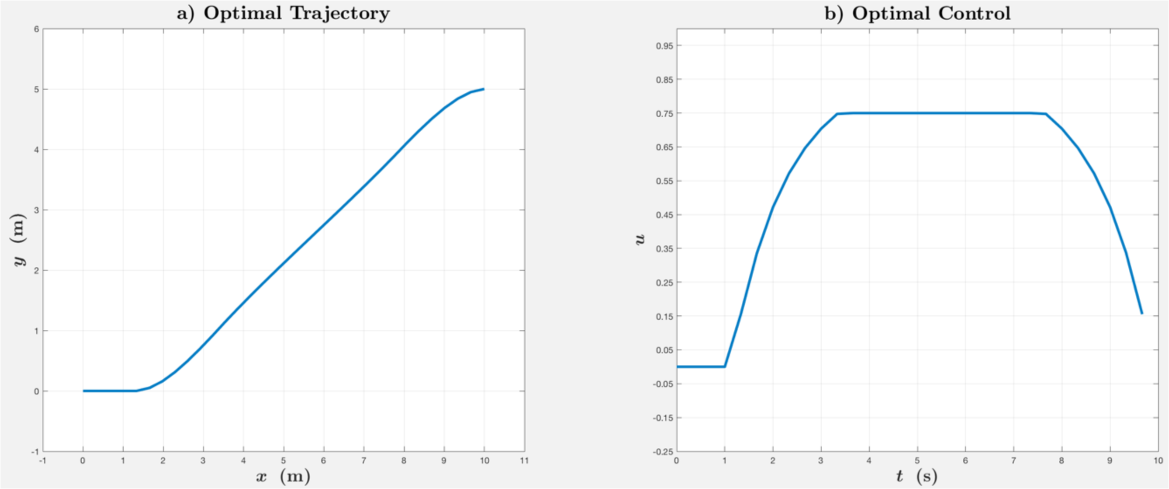
$$x_N = x_0 + h \sum_{i=0}^{N-1} \big(v\cos(u_i) + \mathrm{flow}(y_i)\big),$$

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

Results



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



$|u(t)| \leq 0.75$

Example: Zermelo's Problem

(OCP)

$$\min \int_0^{t_f} u(t)^2 \, dt$$
$$\dot{x}(t) = v\cos\big(u(t)\big) + \mathrm{flow}\big(y(t)\big),\ t \in [0, t_f]$$
$$\dot{y}(t) = v\sin\big(u(t)\big),\ \ t \in [0, t_f]$$
$$(x, y)(0) = 0,\ \ (x, y)(t_f) = (M, \ell)$$
$$|u(t)| \le u_{max},\ \ t \in [0, t_f]$$

(NLOP)

$$\min_{(x_i, u_i)} \sum_{i=0}^{N-1} h\, u_i^2$$
$$x_{i+1} = x_i + h\big(v\cos(u_i) + \mathrm{flow}(y_i)\big)$$
$$y_{i+1} = y_i + h\, v\sin(u_i)\ ,\ |u_i| \le u_{max}$$
$$(x_0, y_0) = 0\ ,\ (x_N, y_N) = (M, \ell)$$

Direct Transcription

(NLOP-C)

$$\min_{u_i} \sum_{i=0}^{N-1} h\, u_i^2$$
$$(x, y)(t_N) = (M, \ell), \qquad |u_i| \le u_{max}$$

where, recursively:

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

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

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

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 $(\mathbf{x}_0(\cdot), \mathbf{u}_0(\cdot))$:

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

3. Solve the new problem $(\textbf{LOCP})_1$ for $(\mathbf{x}_1(\cdot), \mathbf{u}_1(\cdot))$

Sequential Convex Programming

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

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 $\left(\mathbf{x}_k(\cdot), \mathbf{u}_k(\cdot)\right)$ at iteration k:

$$
\mathbf{f}_{k+1}(\mathbf{x}, \mathbf{u}, t) = \mathbf{f}(\mathbf{x}_k(t), \mathbf{u}_k(t), t) + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}_k(t), \mathbf{u}_k(t), t)(\mathbf{x} - \mathbf{x}_k(t)) + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{x}_k(t), \mathbf{u}_k(t), t)(\mathbf{u} - \mathbf{u}_k(t))
$$

and solve the problem $(\textbf{LOCP})_{k+1}$ for $\left(\mathbf{x}_{k+1}(\cdot), \mathbf{u}_{k+1}(\cdot)\right)$

Sequential Convex Programming

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

Discretize and solve a convex problem at each iteration

1. Select a discretization $0 = t_0 < t_1 < \cdots < t_N = t_f$ 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 $h_i = t_{i+1} - t_i$, $(\textbf{LOCP})_{k+1}$ is transcribed into the following convex optimization problem

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

Sequential Convex Programming

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

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

SCP Methodology: at each iteration k,

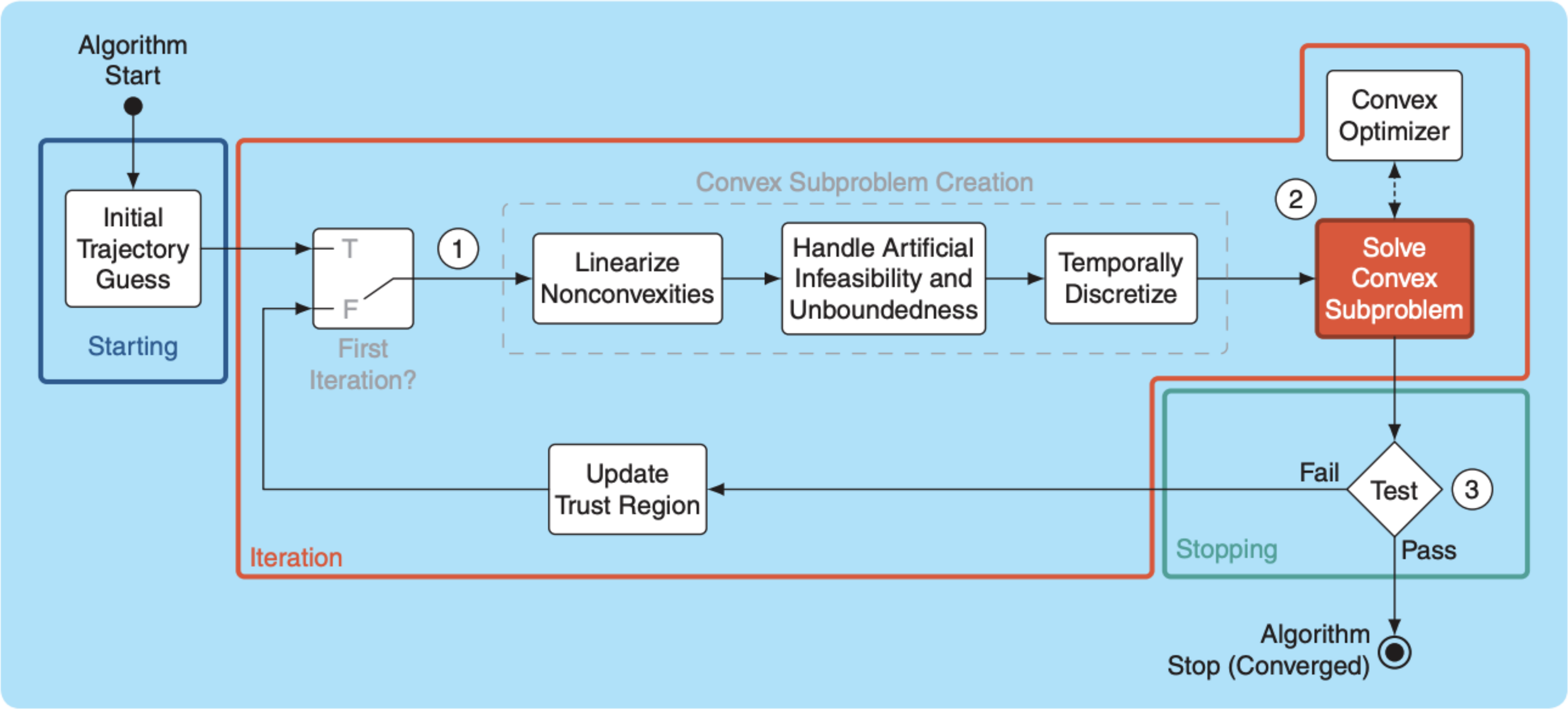
Linearize $\mathbf{f}$ around the solution $\left(\mathbf{x}_k(\cdot), \mathbf{u}_k(\cdot)\right)$

→ Define the continuous time problem $(\textbf{LOCP})_{k+1}$

↓ Discretize $(\textbf{LOCP})_{k+1}$ in time and define the convex optimization problem $(\textbf{DLOCP})_{k+1}$

← Solve $(\textbf{DLOCP})_{k+1}$ via convex programming for a discretized version of $\left(\mathbf{x}_{k+1}(\cdot), \mathbf{u}_{k+1}(\cdot)\right)$

Sequential Convex Programming



For more info: D. Malyuta *et al.*, "Convex Optimization for Trajectory Generation: A Tutorial on Generating Dynamically Feasible Trajectories Reliably and Efficiently," in *IEEE Control Systems Magazine*, vol. 42, no. 5, pp. 40-113, Oct. 2022.

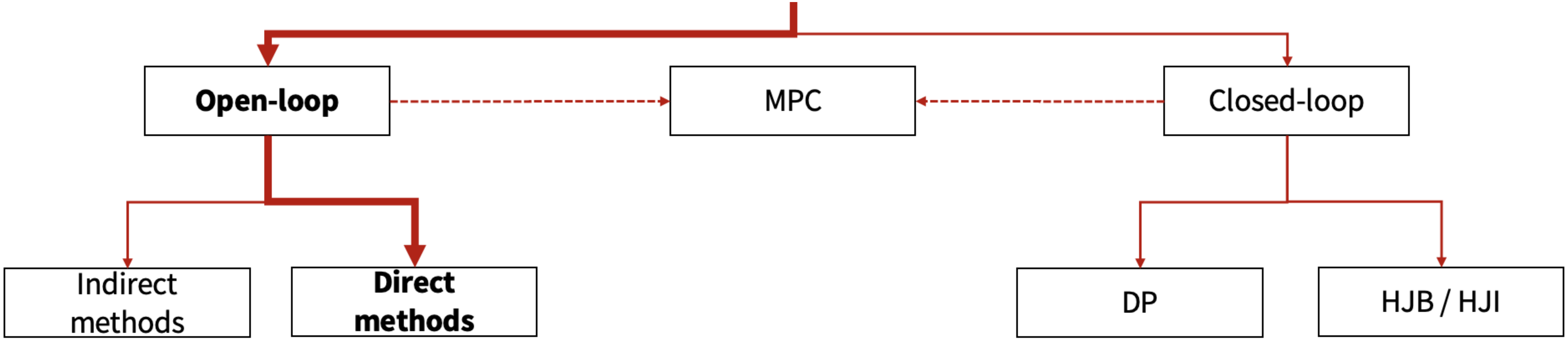
Direct Methods in Practice

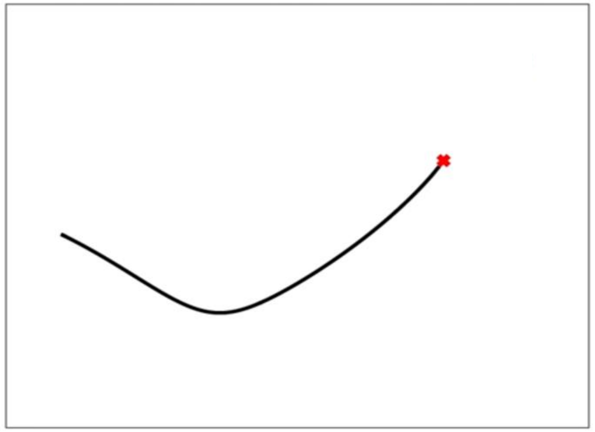
“As you begin to play with these algorithms on your own problems, you might feel like you're on an emotional roller-coaster.” – Russ Tedrake

- Better initial guess trajectories (“warm-starting” the optimization, as seen in `zermelo_simultaneous`)

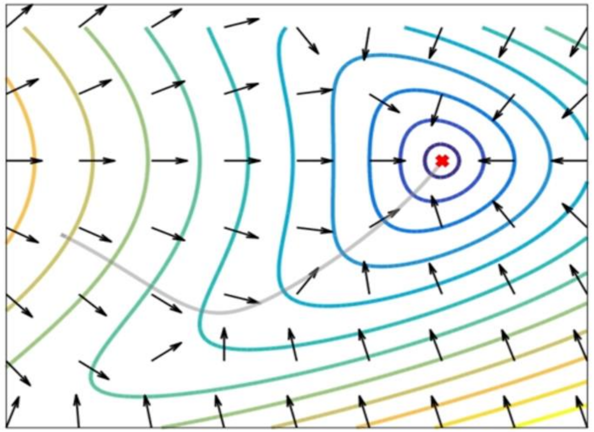
- Cost function/constraint tuning (as seen in `zermelo_scp`)
  - Penalty methods; augmented Lagrangian-based solvers

To wrap up...





- Local vs Global solution
- Less vs more compute
- High- vs low-dimensional system



Next time

- Dynamic programming
- Discrete LQR