AA/EE/ME 548: Linear Multivariable Control

Lecture 13

5/12/2025

Announcements

- Homework 2 due Thursday
  - Homework 2 solution folks, get ready
- Homework 3 out
  - Shorter + project updates
- Week 6 folks, worked examples due Wednesday!
- May 19th lecture: Mega OH with Oliver!
  - Homework and project!

Last week

- Linear Quadratic Regulator
  - Set up LQR problem + plug into Bellman/HJB equation
- Discrete-time, finite-horizon
  - Set $P_T = Q_T$
  - (Offline) For $t = T, T-2, \ldots, 2,1$:
    - $P_{t-1} = Q + A^T P_t A - A^T P_t B (R + B^T P_t B)^{-1} B^T P_t A$
    - $K_{t-1} = (R + B^T P_t B)^{-1} B^T P_t A$
  - Running controller online, $t = 0, \ldots, T-1$: $u_t^* = -K_t x_t$
- Discrete-time, infinite horizon
  - $P_{ss} = Q + A^T P_{ss} A - A^T P_{ss} B (R + B^T P_{ss} B)^{-1} B^T P_{ss} A$
    - Discrete Algebraic Riccati Equation (DARE)

Last week

- Linear Quadratic Regulator
  - Set up LQR problem + plug into Bellman/HJB equation
- Continuous-time, finite-horizon
  - Set $P(T) = Q_T$
  - (Offline) Solve Riccati equation:
    - $\dot{P} + Q - PBR^{-1}B^TP + A^TP + PA = 0$
  - Running controller online, $K(t) = R^{-1}B^TP(t),\ u_t^* = -K_t x_t$
- Continuous-time, infinite horizon
  - $Q - PBR^{-1}B^TP_{ss} + A^TP_{ss} + P_{ss}A = 0$
    - Continuous Algebraic Riccati Equation (CARE)

Last week

- Tracking LQR
  - Nominal trajectory: $(\bar{x}_0, \bar{x}_1, \ldots, \bar{x}_T), (\bar{u}_0, \bar{u}_1, \ldots, \bar{u}_T)$
  - Error dynamics: $\delta x_{t+1} = A\delta x_t + B\delta u_t$
  - Apply LQR on *error dynamics*
  - $u_t^* = \bar{u}_t - K_t \delta x_t$

This week

- Iterative LQR (iLQR)
- Trajectory optimization
  - Sequential Convex Programming (Oliver)
  - Direct methods
    - Shooting method
    - Simultaneous method
    - Psuedospectral method
    - Differential flatness
    - Model Predictive Path Integral (MPPI)

https://www.matthewpeterkelly.com/tutorials/trajectoryOptimization/index.html

Tracking LQR continued…

- Consider the standard LQR tracking problem
  - Nominal trajectory: $(\bar{x}_0, \bar{x}_1, \ldots, \bar{x}_T), (\bar{u}_0, \bar{u}_1, \ldots, \bar{u}_T)$
- But dynamics are nonlinear
  - $x_{t+1} = f(x_t, u_t, t)$
- Need to make dynamics *linear*
  - Apply first order approximation!

HW1: Using JAX, getting all the As and Bs along all points along a trajectory is a "one-liner"

- $\delta x_{t+1} = x_{t+1} - f(\bar{x}_t, \bar{u}_t) \approx \frac{\partial f}{\partial x}(\bar{x}_t, \bar{u}_t)^T \delta x_t + \frac{\partial f}{\partial u}(\bar{x}_t, \bar{u}_t)^T \delta u_t$
  - Error dynamics: $\delta x_{t+1} = A_t \delta x_t + B_t \delta u_t$
- *Time-varying* LQR
- Potential issues?

(Simple) Nonlinear optimal control problem

$$\min_{\mathbf{u}} \sum_{t=0}^{T-1} c(x_t, u_t) + c_T(x_T)$$

$$\text{subject to } x_{t+1} = f(x_t, u_t, t)$$

$$x_0 = x_{\text{curr}}$$

- Nonlinear dynamics
- Non-quadratic cost
- How to solve this nonlinear OCP?

Iterative LQR

- Feasible trajectory (initial guess) $\tau^0 = (\bar{x}_0^0, \bar{x}_1^0, \dots, \bar{x}_T^0), (\bar{u}_0^0, \bar{u}_1^0, \dots, \bar{u}_{T-1}^0)$
- Linearize dynamics about τ^0: $x_{t+1} \approx A_t^0 \, x_t + B_t^0 u_t + C_t^0, \, t = 0, .., T-1$
- Second-order approximation on cost ("quadratize") about τ^0
  - $c(x_t, u_t) \approx c(\bar{x}_t, \bar{u}_t) +$
    $c_x(\bar{x}_t, \bar{u}_t)^T \delta x_t + c_u(\bar{x}_t, \bar{u}_t)^T \delta u_t +$
    $\frac{1}{2} \delta x_t^T c_{xx} \delta x_t + \frac{1}{2} \delta u_t^T c_{uu} \delta u_t + \delta x_t^T c_{xu} \delta u_t$

Dropping the super script…

$$\approx \tilde{c}_t + \tilde{q}_t^T \delta x_t + \tilde{r}_t^T \delta u_t + \frac{1}{2} \delta x_t^T \tilde{Q}_t \delta x_t + \frac{1}{2} \delta u_t^T \tilde{R}_t \delta u_t + \delta x_t^T \tilde{N}_t \delta u_t$$

Expand the δx_t and δu_t terms and group constants, linear, and quadratic terms w.r.t x_t and u_t

$$c_t \approx c_t + q_t^T x_t + r_t^T u_t + \frac{1}{2} x_t^T Q_t x_t + \frac{1}{2} u_t^T R_t u_t + x_t^T N_t u_t$$

Iterative LQR

- Feasible trajectory $\tau^0 = (\bar{x}_0^0, \bar{x}_1^0, \ldots, \bar{x}_T^0), (\bar{u}_0^0, \bar{u}_1^0, \ldots, \bar{u}_{T-1}^0)$
- Linearize dynamics about τ^0:
  - $A_t^0, B_t^0, C_t^0, t = 0, .., T-1$
- Second-order approximation on cost ("quadratize") about τ^0
  - $c_t^0, q_t^0, r_t^0, Q_t^0, R_t^0, N_t^0, , t = 0, .., T-1$
- Solve LQR problem (slightly more complex with linear and cross terms and time-varying terms)
- Get u_t^* and simulate to get $\tau^1 = (\bar{x}_0^1, \bar{x}_1^1, \ldots, \bar{x}_T^1), (\bar{u}_0^1, \bar{u}_1^1, \ldots, \bar{u}_{T-1}^1)$
- Then repeat!

iLQR algorithm

- Nominal state & controls $\tau^0 = (\bar{x}_0^0, \bar{x}_1^0, \ldots, \bar{x}_T^0), (\bar{u}_0^0, \bar{u}_1^0, \ldots, \bar{u}_{T-1}^0)$

For $k = 0, \ldots, N$

1. Linearize dynamics about τ^k
2. Quadratize cost about τ^k
3. Solve LQR problem
4. Use optimal control to get corresponding states (use nonlinear dynamics)
5. Obtain $\tau^{k+1} = (\bar{x}_0^k, \bar{x}_1^k, \ldots, \bar{x}_T^k), (\bar{u}_0^k, \bar{u}_1^k, \ldots, \bar{u}_{T-1}^k)$

Algorithmic considerations

- Linearization and "quadratization" valid only for a local region around nominal post
  - Introduce a "trust region". Penalize large deviations from τ^{k-1}
  - $c_{TR}\left(\delta x_t^k, \delta u_t^k\right) = \delta x_t^{k^T} Q_{TR} \delta x_t^k + \delta u_t^{k^T} R_{TR} \delta u_t^k$
- Cost matrices may not be PSD/PD
  - Make negative eigenvalues 0
  - Add λI to the matrices for sufficiently large λ
- Local method. May get stuck in local optimum
- Termination criteria
- Choosing initial trajectory

Trajectory optimization (direct methods)

MIT Underactuated Robotics (Chapter 10)

Matthew Kelly Trajectory Optimization notes

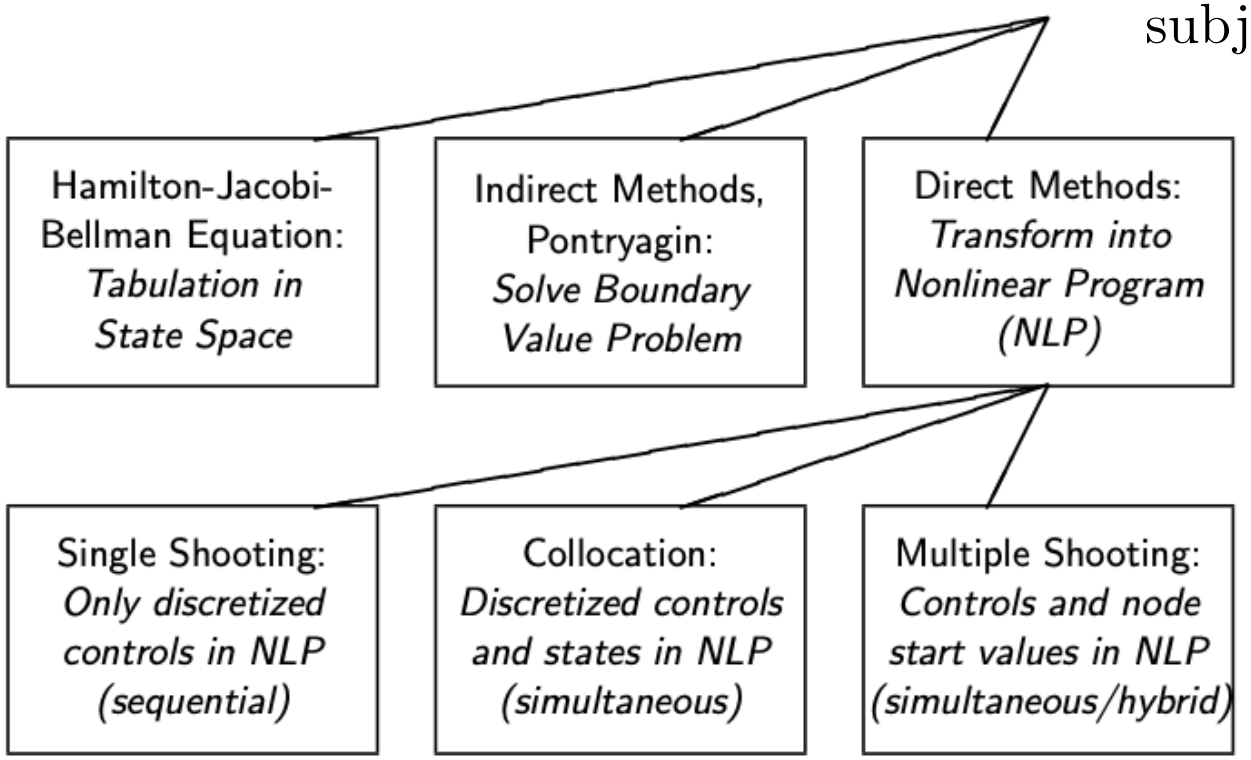
General trajectory optimization problem

$$\min_{\mathbf{u}} \quad \sum_{t=0}^{T-1} J(x_t, u_t, t) + J_T(x_T)$$

$$\text{subject to} \quad x_{t+1} = f(x_t, u_t, t), t = 0, ..., T-1$$

$$g(\mathbf{x}, \mathbf{u}) \leq 0$$

$$h(\mathbf{x}, \mathbf{u}) = 0$$



Moritz Diehl notes

Direct methods: Transform into NLP

- Just "type it out" and "plug it into a solver"!
- Easier said than done
  - The optimization landscape needs to be "nice"
  - Differentiability
  - Accuracy vs speed
  - Discrete time; what about in between time steps?
  - Is the solution we get *the* optimal solution?
  - Different ways to "massage" and frame the problem
    - A bit of an art

The "art" of trajectory optimization (direct methods)

Turning

$$\begin{aligned} \min_{\mathbf{u}} \quad & \sum_{t=0}^{T-1} J(x_t, u_t, t) + J_T(x_T) \\ \text{subject to} \quad & x_{t+1} = f(x_t, u_t, t), t = 0, ..., T-1 \\ & g(\mathbf{x}, \mathbf{u}) \leq 0 \\ & h(\mathbf{x}, \mathbf{u}) = 0 \end{aligned}$$

Into something we can tractably solve

1. *Solving* the optimization problem (convexification)
2. *Formulating* the optimization problem

Examples of Convex Optimization for Control

- Since powerful off-the-shelf convex optimizers now exist, there's been a big push toward *convexifying* real-world problems.
- In other words: since solving convex problems is (usually) "easy", for many applications the hard part is now *figuring out how to represent real-world problems in a convex way*.
- Some examples…

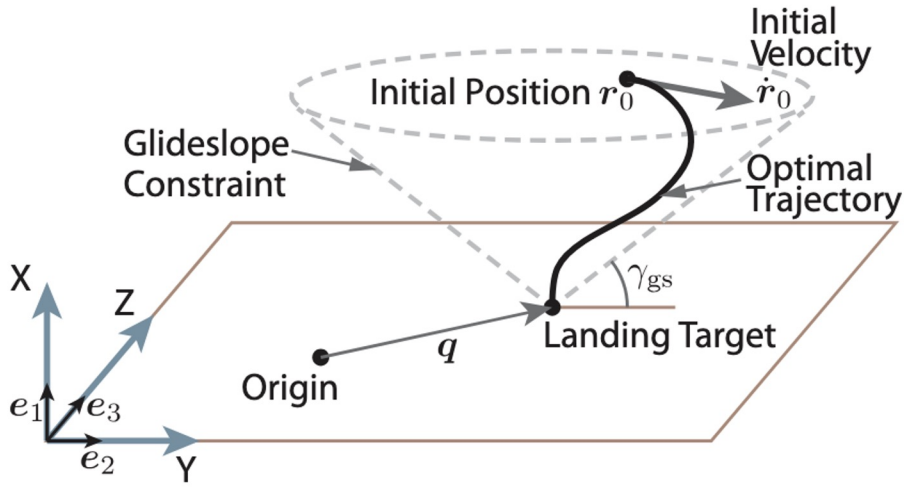
Sequential Convex Programming

- Idea: Convexify problem about nominal solution, and then solve
- Similar to the iLQR algorithm, but more general
  - Can handle constraints
- Sequential *Quadratic* Programming
  - Turn problem into *Quadratic* Program (a nice convex program)
- Sequential *Convex Programming*
  - Turn problem into *convex* program.
  - Need more "tricks"
  - Perform theoretical analysis to offer insights into convergence properties, optimality of solution
- Sensitive to initial guess
  - Work on warm-starting

Example: Lossless Convexification

Behçet Açıkmeşe, Lars Blackmore http://larsblackmore.com/iee_tcst13.pdf

Setup: rocket landing



Problem: not just *max* thrust, but *min* thrust constraints (nonconvex!)

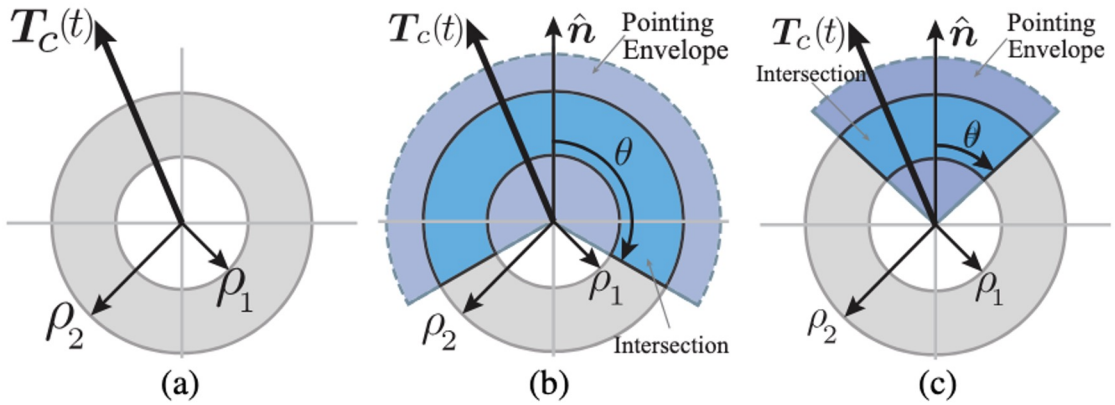
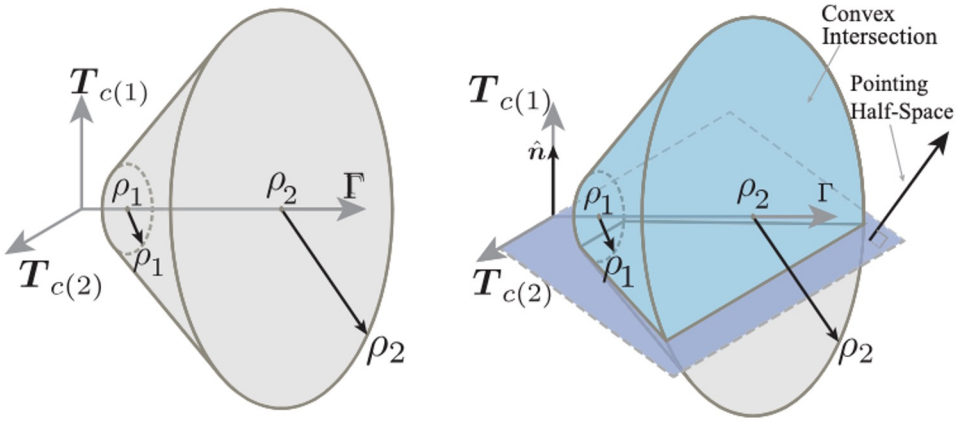


Fig. 2. (a) Planar representation of original thrust bounds, intersection of thrust bounds and thrust pointing limits: (b) $\theta \in (\pi/2, \pi)$ and (c) $\theta \in [0, \pi/2]$.

Example: Lossless Convexification

Behçet Açıkmeşe, Lars Blackmore http://larsblackmore.com/iee_tcst13.pdf

Solution: raise control variable into higher-dimensional convex set…



… and prove that these two (different!) control constraints give *the same result at optimality.*

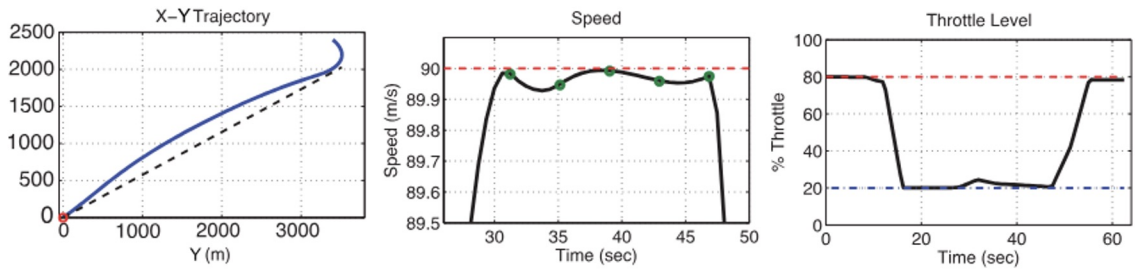


Fig. 7. Example with one contact with the glide slope boundary and two contacts with the speed boundary. The throttle and thrust angle from vertical time profiles (second row) indicate that the thrust vector profile is feasible for Problems 1 and 2, and the convexification still holds, and we obtain an optimal solution to the soft landing problem.

Example: Hypersonic Reentry

Skye Mceowen et al. (link)

Setup: hypersonic reentry guidance (nonconvex dynamics, nonconvex obstacle constraints).

OK, just use successive convex programming, right?

Problem: numerical tradeoffs

- Want more discretization points for accuracy, but less for compute performance
- Long-range trajectory with highly nonlinear dynamics hurts accuracy

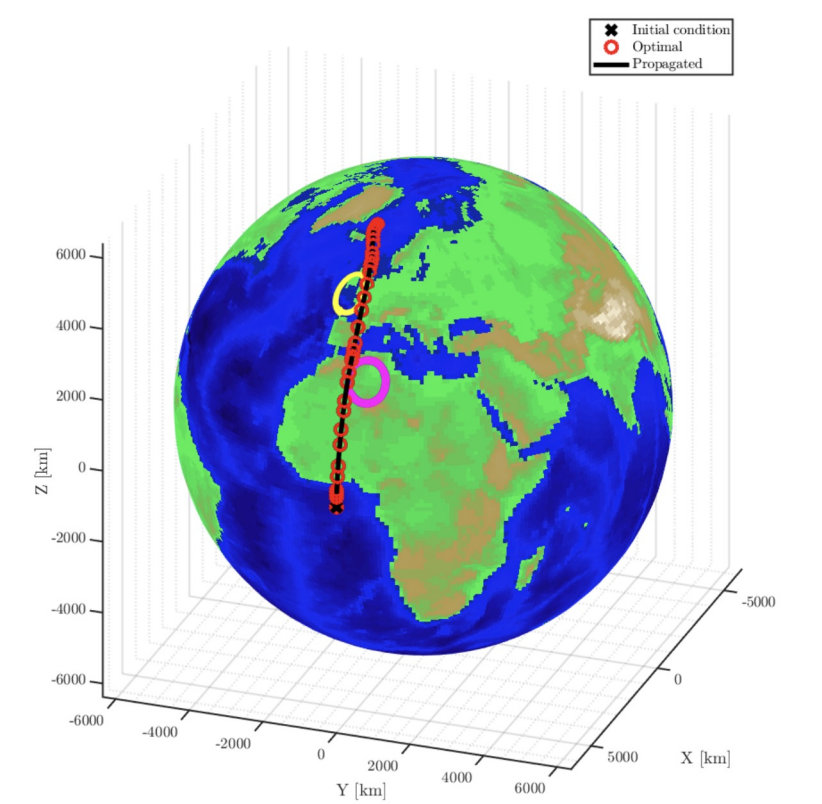


Fig. 15 **A 3-DoF solution trajectory for the spherical Earth problem with NFZs.**

Example: Hypersonic Reentry

Skye Mceowen et al. (link)

Solution: explicitly consider linearization and discretization error…

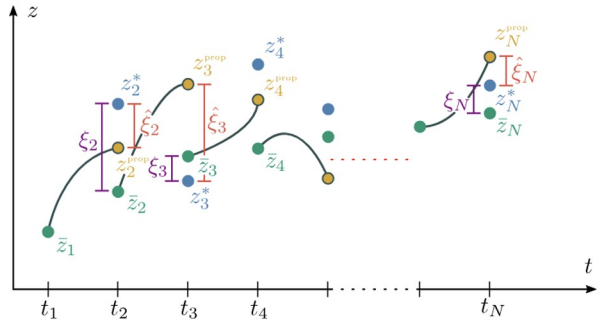


Fig. 3 Variables introduced into the stitching constraints are depicted. Each continuous trajectory segment represents a piecewise integration across each timestep used in the exact discretization, where $\bar{z}_k$ represents the initial condition and z_{k+1}^{prop} represents the final condition.

… and modify SCP algorithm to drive this to zero during trajectory optimization

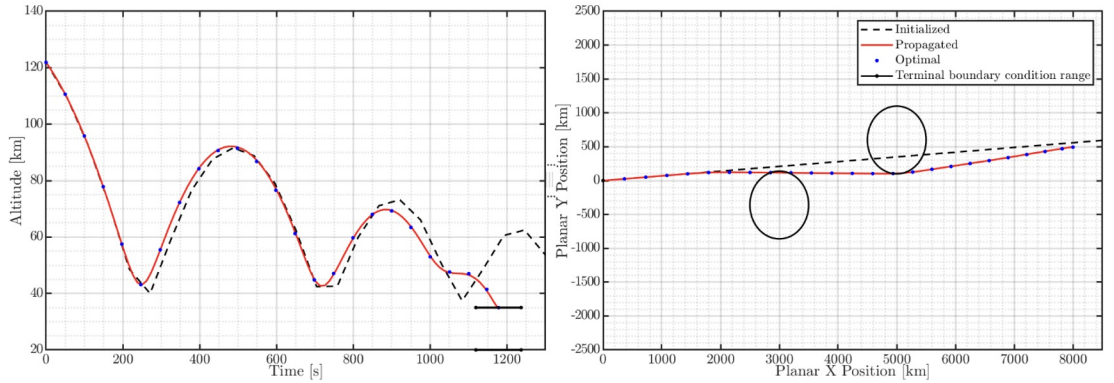


Fig. 5 Left: Altitude solutions are displayed for the planar Earth problem. Right: Planar position for the planar Earth problem, avoiding NFZs.

Open-loop continuous nonlinear trajectory matches discretization almost exactly!

Example: Deferred-Decision Traj. Opt.

Purnanand Elango et al. (link)

Setup: basic trajectory optimization (get from point A to point B), not necessarily nonconvex; so what's the big deal?

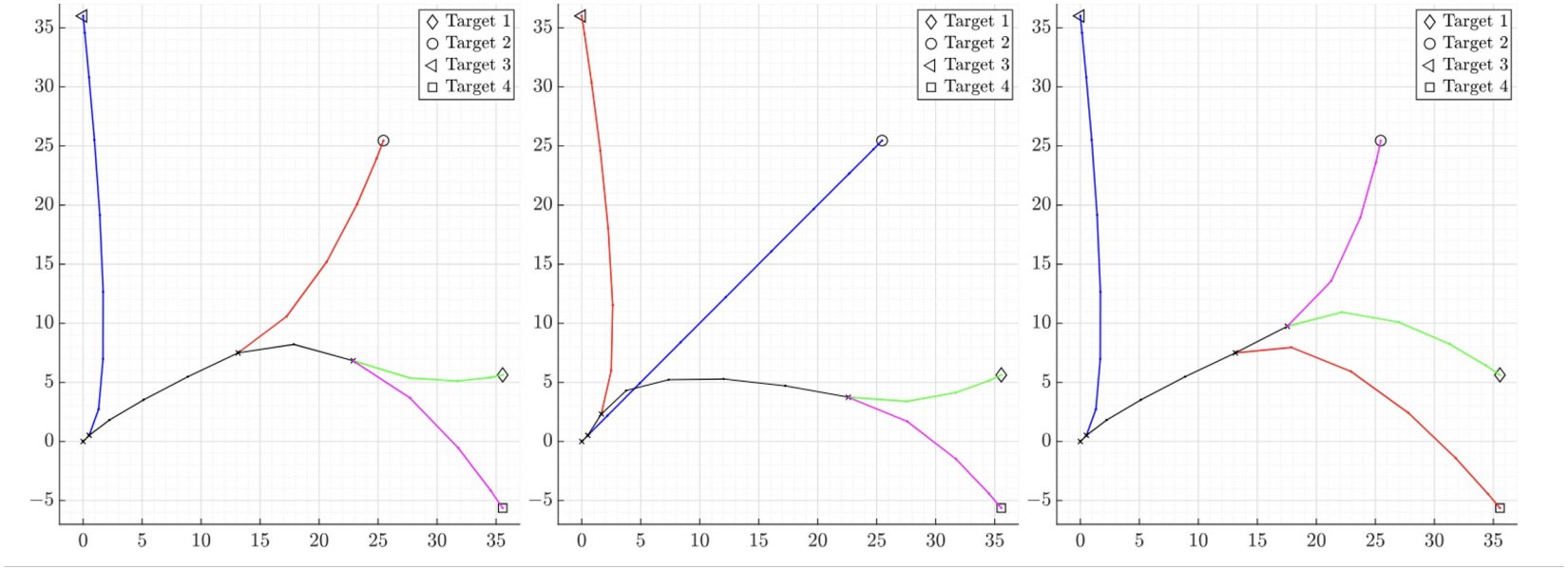
Problem: what if we don't actually know point B when we start the trajectory? What if there are several destinations we might choose from, and we don't know which we prefer until we actually execute the trajectory?

For example, only choosing one of a few potential landing sites once you're through the clouds and can see them better.

Example: Deferred-Decision Traj. Opt.

Purnanand Elango et al. (link)

Solution: plan trajectories to all potential terminal points at once, and maximize the length of time that the trajectories remain identical after they start.



This allows you to *defer* the decision of which terminal point to go to until you're actually "in flight".

Example: Robust Linear Traj. Opt.

Oliver Sheridan, Behçet Açıkmeşe (link, UW students can sign in to access)

Setup: discrete-time linear trajectory optimization problem, with uncertainty

$$x_{t+1} = A_t x_t + B_t u_t + E_t n_t.$$

Unknown, but known bounds

Problem: how can we plan a trajectory that we know will satisfy our state constraints when there's uncertainty in the system?

Convex solvers won't handle random variables for you!

Example: Robust Linear Traj. Opt.

Oliver Sheridan, Behçet Açıkmeşe (link, UW students can sign in to access)

Solution: analytically bound maximum state deviation from nominal, derive deterministic convex constraint equivalent to robust satisfaction of original constraint

Theorem 1: If strong duality holds for the problems (6), and if $\bar{K}_{t-1}$ is fixed, the robust state constraint (5c) is equivalent to the following:

$$\Gamma_t \bar{f}\left(\bar{x}_{t-1}^n, \bar{v}_{t-1}\right) \leq M_{\mathrm{RHS}}, \tag{9}$$

where

$$\Gamma_t = \max\left(\mathbf{0}, H_t\left(\mathbf{I} + \bar{B}_{t-1}\bar{K}_{t-1}\right)\bar{E}_{t-1}\right).$$

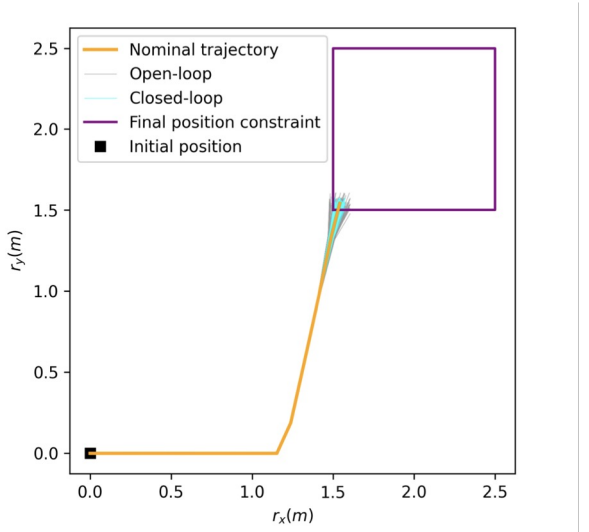


Fig. 1. Overview of nominal trajectory and perturbed Monte Carlo runs.

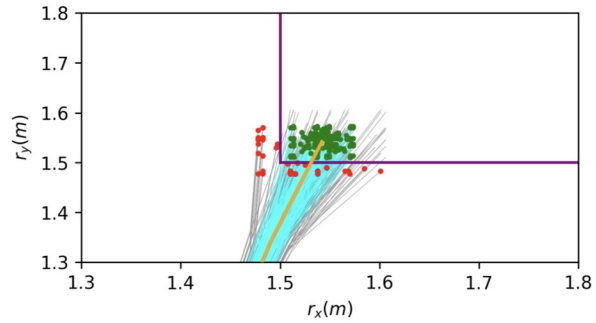


Fig. 2. Close-up of final position.

Direct methods

- **Direct transcription:** "First discretize, then optimize"
  - Turn infinite dimensional problem (i.e., continuous time problem) into a finite dimensional problem (i.e., discrete time)
  - Solve finite dimension problem as NLP

- Leverage state-of-the-art methods for solving NLP
- Approximate solution to continuous time problem

Shooting method: Cannon example

Find: $\mathbf{z}(t) = [x(t), y(t), \dot{x}(t), \dot{y}(t)]$

Boundry Conditions:

$$x(t_0) = 0$$
$$y(t_0) = 0$$
$$x(t_f) = x_T$$
$$y(t_f) = y_T$$

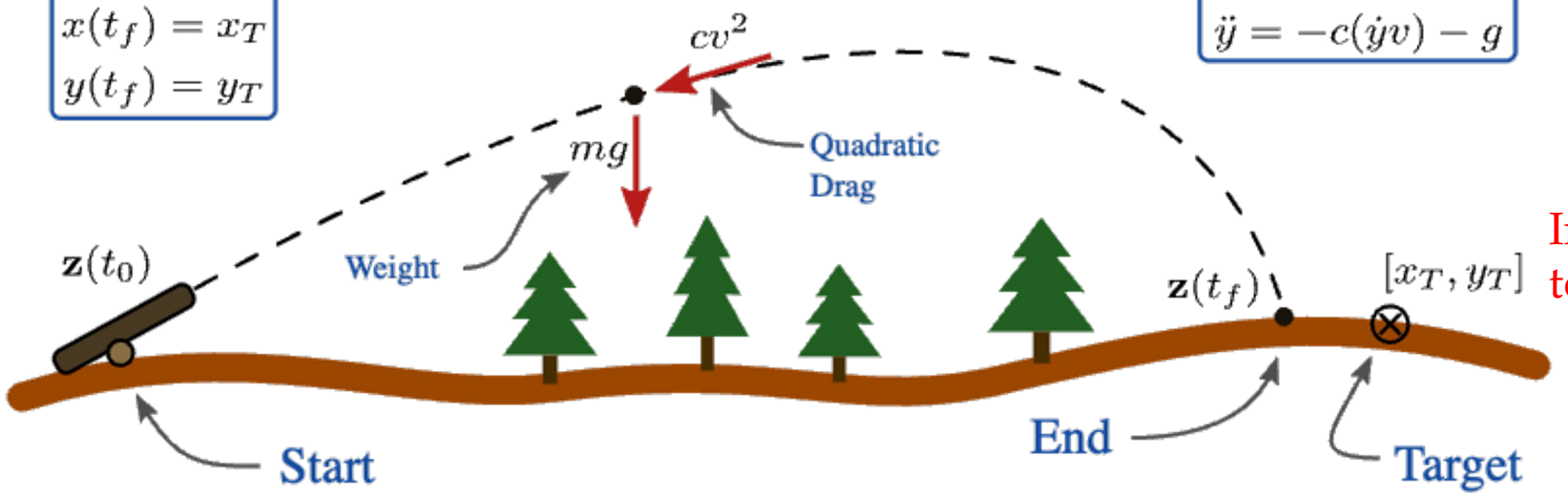
Cost Function:

$$J = \dot{x}(t_0)^2 + \dot{y}(t_0)^2$$

Dynamics:

$$v = \sqrt{\dot{x}^2 + \dot{y}^2}$$
$$\ddot{x} = -c(\dot{x}v)$$
$$\ddot{y} = -c(\dot{y}v) - g$$

State constraints

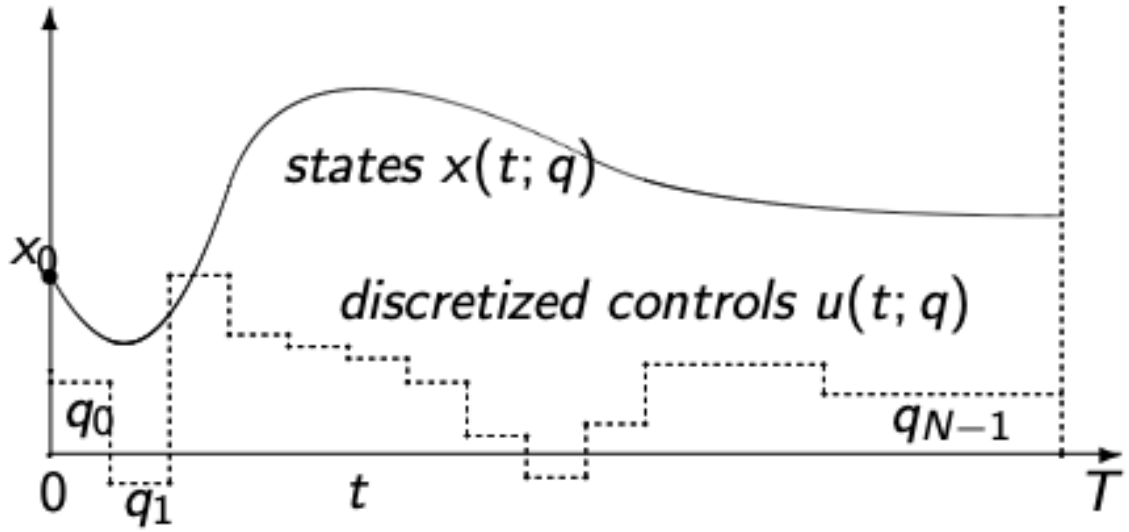


Integrate dynamics to get final state

Control: Initial angel and speed

Matthew Kelly Trajectory Optimization notes

Shooting method (general)



Decision variables: Controls $u_0, u_1, \ldots, u_{T-1}$

Get states by integrating dynamics using $(x_0, u_0, u_1, \ldots, u_{T-1})$

Moritz Diehl notes

Shooting method (Discussion)

- Optimize only for control sequence. Need to *simulate dynamics* to get states
- Less variables
- Can be effective on short horizon, well-behaved dynamics
- Solution will be *dynamically feasible*
- Sensitive to initial guess
- Numerical conditioning for long horizons (especially for unstable or stiff systems)
  - Vanishing gradients
  - Earlier controls have less impact than later controls
- Difficult to enforce state constraints directly. Need to ”unroll” states first
- Inherently a serial operation