Four Lectures on Polynomial Optimization for Control

Lecture 4: Robust motion planning using SOS methods

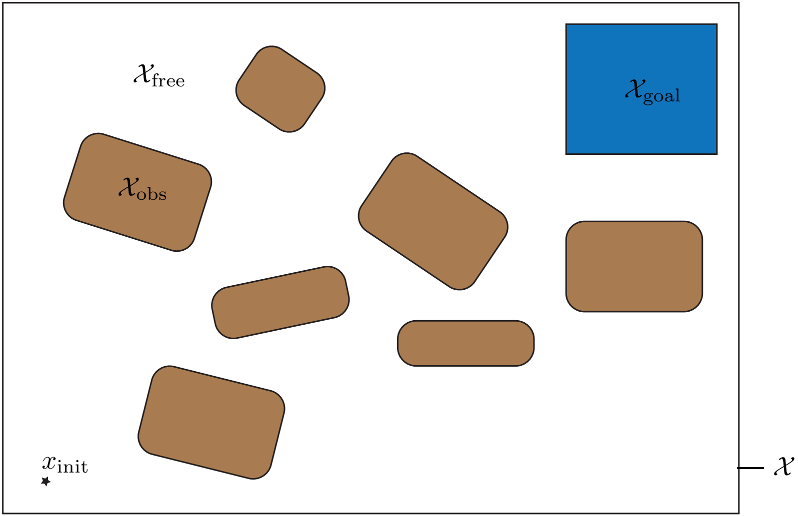
Joe Warrington

Motivation

- **Complex robotic tasks** cannot be accomplished with simple static feedback controllers. Example applications:
  - Route planning through an obstacle-strewn environment
  - Aircraft collision avoidance
  - Robot arm trajectories
- **Difficulties**
  - Nonlinear dynamics
  - Non-convex constraints
  - Model uncertainty
  - Task differs between instances
- **Typical scenario**
  - Plan a nominal trajectory according to some criterion: minimum time, cost, or just feasibility
  - Implement a mixture of feed-forward control and local state feedback to track nominal trajectory

Motion planning problem

- Initial state x_{init}, goal set $\mathcal{X}_{\text{goal}}$, free space $\mathcal{X}_{\text{free}} := \mathcal{X} \backslash \mathcal{X}_{\text{obs}}$
- Task is to find a minimum-cost path from x_{init} to any $x \in \mathcal{X}_{\text{goal}}$, passing only through $\mathcal{X}_{\text{free}}$.



Motion planning: Single- and multi-query

- In a **single-query** application, we do not attempt to re-use previous planning solutions. Each time a plan is required, a new one is constructed from scratch.
- In a **multi-query** application, one round of computation is performed offline, and used multiple times online for different motion tasks.

- Most widely used motion planning algorithms:
  - Single query: **RRT (rapidly exploring random trees)**
  - Multiple query: **PRM (probabilistic road maps)**

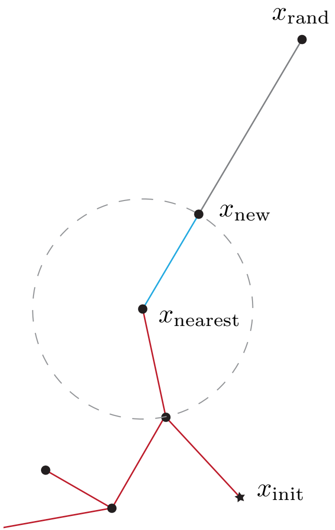
- Many variants of both algorithms exist
- Focus of this lecture: **RRT**

Motion planning: RRT

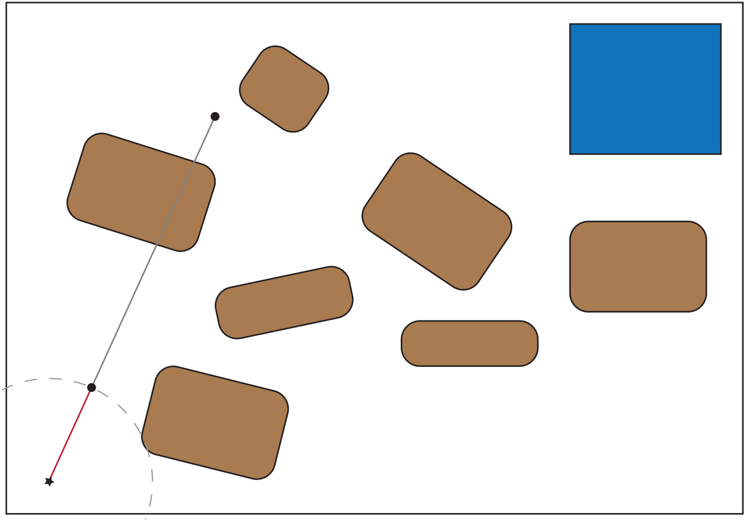
Algorithm:

$\mathcal{V} \leftarrow \{x_{\text{init}}\}$
$\mathcal{E} \leftarrow \emptyset$
for $i = 1$ **to** n **do**
  $x_{\text{rand}} \leftarrow$ `SampleFreeSpace`
  $x_{\text{nearest}} \leftarrow$ `NearestNode`$(\mathcal{V}, x_{\text{rand}})$
  $x_{\text{new}} \leftarrow \Pi_{\mathbf{B}(x_{\text{nearest}})}(x_{\text{rand}})$
  if `ObstacleFree`$(x_{\text{nearest}}, x_{\text{new}})$ **then**
    $\mathcal{V} \leftarrow \mathcal{V} \cup \{x_{\text{new}}\}$
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{(x_{\text{nearest}}, x_{\text{new}})\}$
  end if
end for
return $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

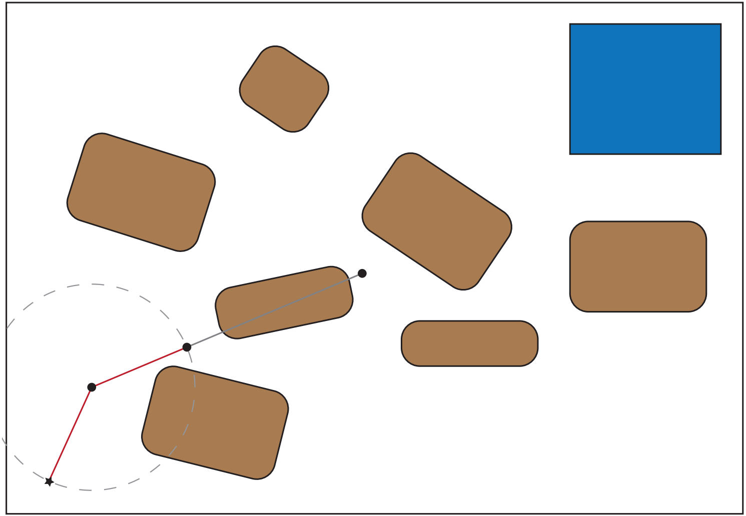
- Can also be terminated when a node in a "goal set" is connected to the tree.



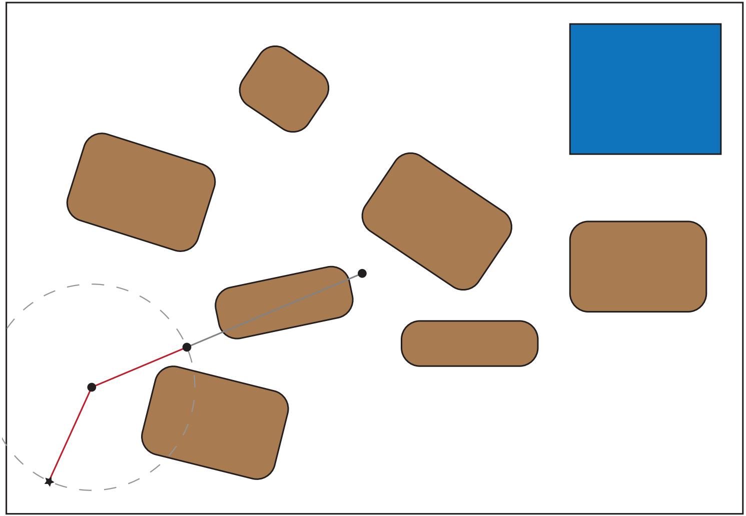
RRT example



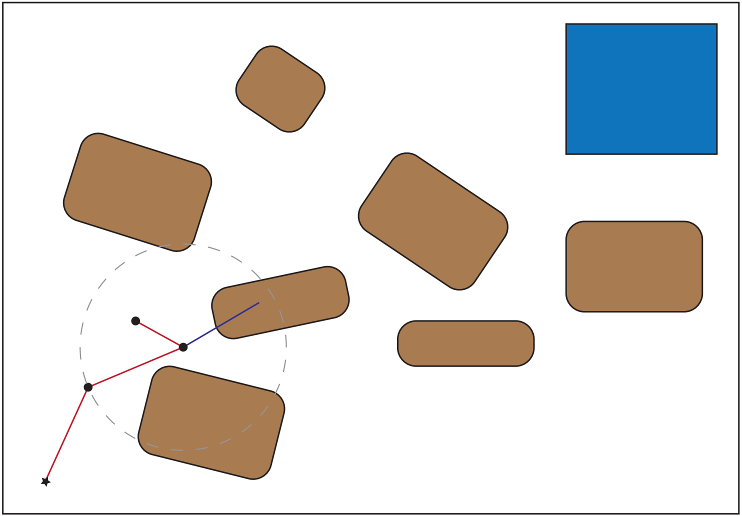
RRT example



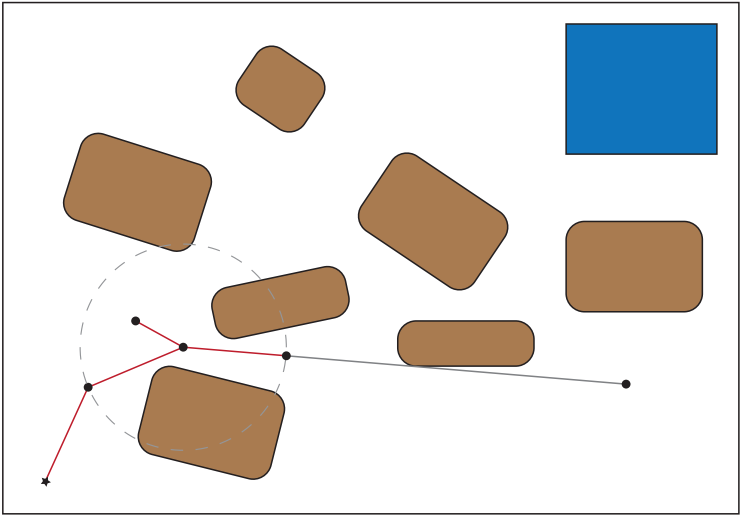
RRT example



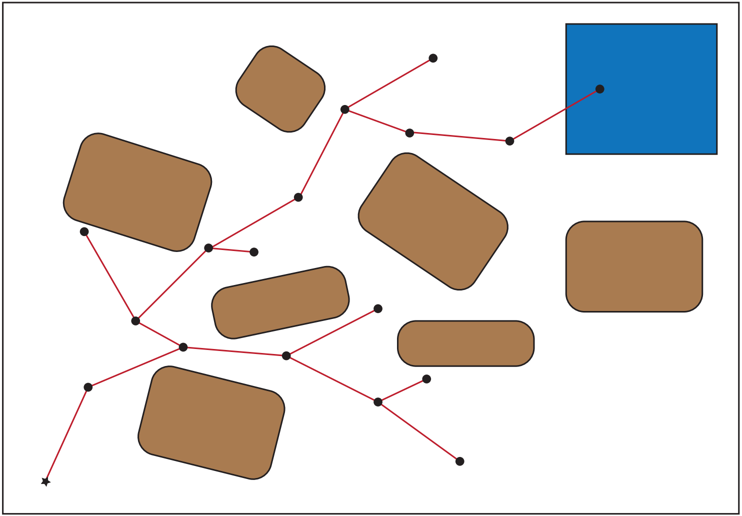
RRT example



RRT example



RRT example



Motion planning: RRT*

- **Revised algorithm:**

$\mathcal{V} \leftarrow \{x_{\text{init}}\}$
$\mathcal{E} \leftarrow \emptyset$
for $i = 1$ **to** n **do**
  $x_{\text{rand}} \leftarrow$ `SampleFreeSpace`
  $x_{\text{nearest}} \leftarrow$ `NearestNode`$(\mathcal{V}, x_{\text{rand}})$
  $x_{\text{new}} \leftarrow \Pi_{\mathbf{B}(x_{\text{nearest}})}(x_{\text{rand}})$
  if `ObstacleFree`$(x_{\text{nearest}}, x_{\text{new}})$ **then**
    $\mathcal{V} \leftarrow \mathcal{V} \cup \{x_{\text{new}}\}$
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{(x_{\text{nearest}}, x_{\text{new}})\}$
    `Rewire`(x_{new})
  end if
end for
return $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- The `Rewire` step looks for lower-cost connections in the neighbourhood of x_{new}, and creates or deletes links such that the result is still a tree.

Property 1: Probabilistic completeness

- A sampling-based algorithm is **probabilistically complete** if for any robustly feasible[1] routing problem,

$$\liminf_{n\to\infty} \mathbb{P}\left(\exists x_{\text{goal}} \in \mathcal{V}_n \cap \mathcal{X}_{\text{goal}} : x_{\text{init}} \text{ is connected to } x_{\text{goal}} \text{ through } \mathcal{G}_n\right) = 1$$

  where $\mathcal{G}_n := (\mathcal{V}_n, \mathcal{E}_n)$ is the vertex set at iteration n of the algorithm.

- RRT is known to be probabilistically complete:

Theorem (Lavalle and Kuffner 2001)

If a path planning problem is robustly feasible then there exist constants $a > 0$ and $n_0 \in \mathbb{N}$ (depending on the planning environment but not x_{init}), such that the RRT algorithm achieves

$$\mathbb{P}(\mathcal{V}_n \cap \mathcal{X}_{\text{goal}} \neq \emptyset) > 1 - e^{-an}, \quad \forall n > n_0.$$

- In other words, the chance of RRT failing to connect x_{init} to the goal set $\mathcal{X}_{\text{goal}}$ decreases exponentially to zero with the number n of iterations, at least after some n_0.

[1] Here, *robustly feasible* means there exists a small enough $\delta > 0$ by which all obstacles can be increased in size such that the problem remains feasible. The condition is used to rule out pathological cases where obstacles almost intersect. It has nothing to do with robust control.

Property 2: Asymptotic optimality

- Define a cost function that maps paths through $\mathcal{G}$ to $\mathbb{R}_+$, and let c^* be the cost of a *robustly optimal solution*[2] to the path planning problem.
- A sampling-based algorithm is **asymptotically optimal** if for any path planning problem with a robustly optimal solution with finite cost c^*,

$$\mathbb{P}(\limsup_{n\to\infty} Y_n = c^*) = 1.$$

  In the above expression, Y_n is the optimal cost after iteration n of reaching $\mathcal{X}_{\mathrm{goal}}$ through the tree $\mathcal{G}_n$.
- It turns out that for any sampling-based algorithm, the probability of converging to c^* is either 0 or 1, i.e., the algorithm will either "almost never" or "almost always" converge to an optimal-cost solution.
- The following properties of RRT and RRT* hold:

Theorem (KF2011, Theorems 33 and 38)

The RRT algorithm is not asymptotically optimal, but RRT is.*

- Other technicalities and minor assumptions apply in order to rule out pathological cases; see [KF2011, Section 4.2].

[2] The definition is technical and relates to robust feasibility of the problem; see [KF2011, Section 4.2] for details.

Feedback motion planning

- Smooth dynamical system $\dot{x} = f(x, u)$, with some equilibrium x_G, u_G such that $f(x_G, u_G) = 0$
- Define $\bar{x} = x - x_G$, $\bar{u} = u - u_G$ and define a local linearization, $\dot{\bar{x}} \approx A\bar{x} + B\bar{u}$.
- The cost of regulating to $(\bar{x} = 0, \bar{u} = 0)$ is

$$J(\bar{x}') := \int_0^\infty \left[\bar{x}^\top Q\bar{x} + \bar{u}^\top R\bar{u}\right] \mathrm{d}t$$

$$\text{where} \quad Q \succeq 0, \quad R \succ 0, \text{ and } \bar{x}(0) = \bar{x}'.$$

- This is a standard LQR problem, whose solution is

$$J^*(\bar{x}) = \bar{x}^\top S\bar{x}$$

where $S \succ 0$ solves the continuous algebraic Riccati equation (CARE),

$$Q - SBR^{-1}B^\top S + SA + A^\top S = 0.$$

The associated optimal state feedback controller is linear:

$$\bar{u}^*(\bar{x}) = -R^{-1}B^\top S\bar{x}$$

LQR region of attraction

- Define the ρ-sublevel set

$$\mathcal{B}_G(\rho) = \{x \,:\, 0 \leq V(x) \leq \rho\},$$

- Then $\mathcal{B}_G(\rho)$ is certified to be a **region of attraction** if, for all $x \in \mathcal{B}_G(\rho)$,
  1. $V(x)$ is positive definite (i.e., strictly > 0 except at x_G, where it is 0)
  2. $\dot{V}(x)$ is negative definite (i.e., strictly < 0 except at x_G, where it is 0)
- If these conditions hold, all initial conditions in $\mathcal{B}_G(\rho)$ converge to x_G (Slotine & Li, 1990).
- We will test these conditions for
  - The **true (polynomial or approximated-by-polynomial) dynamics**
  - The linear controller designed for the linearized system around (x_G, u_G)
  - The value function $J(\bar{x}) = \bar{x}^\top S \bar{x}$, acting as a candidate Lyapunov function V
- Insert linear controller into original dynamics:

$$\dot{V}(x) = \dot{J}(\bar{x}) = 2\bar{x}^\top S f(x_G + \bar{x}, u_G - K\bar{x})$$

where $K = -R^{-1}B^\top S$ as in previous slide.
Polynomial condition $\Rightarrow$ use sum of squares programming.

SOS program for largest region of attraction

- To find largest "radius" ρ for which Lyapunov function can be certified, solve the following:

$$\begin{aligned} \max_{\rho, h(\cdot)} \quad & \rho \\ \text{s. t.} \quad & \dot{J}(\bar{x}) + h(\bar{x})(\rho - J(\bar{x})) \leq \epsilon \|\bar{x}\|_2^2 \\ & h \in \Sigma[x]_d, \end{aligned}$$

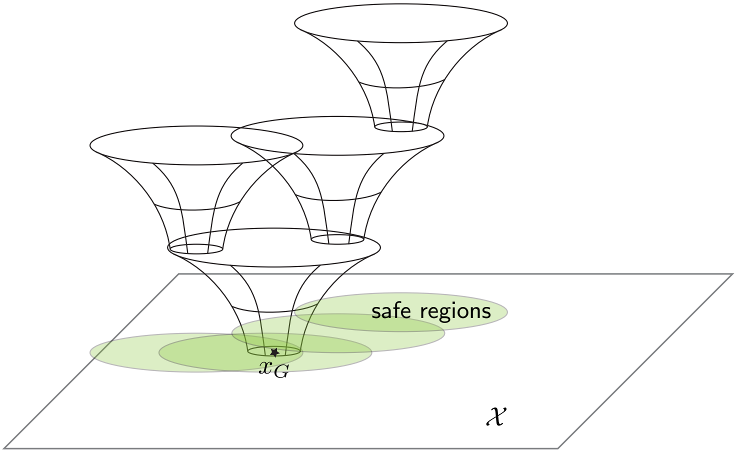
  where the degree d is large enough to be able to match the monomial coefficients found in $\dot{J}(\cdot) = \frac{\partial J}{\partial x} \cdot f(\cdot)$.
- Polynomial $h(\cdot)$ is a multiplier for the constraint $J(\bar{x}) \leq \rho$.
- Problem is **not convex** if ρ and $h(\cdot)$ are both optimization variables, so solve by bisection on ρ. Search for any feasible solution $h(\cdot)$ with $\rho \geq 0$ fixed:

$$\begin{aligned} \min_{h(\cdot)} \quad & 0 \\ \text{s. t.} \quad & \dot{J}(\bar{x}) + h(\bar{x})(\rho - J(\bar{x})) \leq \epsilon \|\bar{x}\|_2^2 \\ & h \in \Sigma[x]_d. \end{aligned}$$

  If infeasible, decrease ρ until it becomes feasible. By smoothness of f, there must exist some small region of attraction around x_G, and a feasible ρ.

Connecting multiple Lyapunov functions

- We wish to combine several such regions of attraction, to cover the state space with "safe" regions, also referred to as **funnels**.
- **Nested Lyapunov functions** (Peterfreund & Baram 1999, Burridge et al. 1999)



Building an LQR-tree

- Any trajectory leading to a point in $\mathcal{B}_G(\rho)$ will ultimately go to x_G.
- Grow a tree in a similar manner to RRT:

1. Pick a point in $x_{\text{new}} \in \mathcal{X} \backslash \mathcal{B}_G(\rho)$ and try to design a finite-time nonlinear trajectory $(x_{\text{nom}}(t), u_{\text{nom}}(t))$ for $t \in [0, t_f]$, where $x(0) = x_{\text{new}}$.
2. End of the trajectory should be on an existing part of the tree.
3. Certify a region of attraction around $x_{\text{nom}}(t)$ for $t \in [0, t_f]$ that covers a non-zero fraction of $\mathcal{X}$
   - Needed because with probability 1 we will never land exactly on the trajectory
   - Therefore need to say something about the surrounding space being safe
4. If no finite time trajectory can be found to connect to the existing tree, discard the point and generate one somewhere else.

- Step 1 is "easy" as mature methods exist: Multiple shooting, collocation methods.
- Step 3 is tricky!

Closed-loop feedback along nonlinear trajectories

- Once a **nonlinear finite-time trajectory** $(x_{\text{nom}}(t), u_{\text{nom}}(t))$ has been found, need to work out how to track or reach it from nearby
- Define a **time-varying linearization** for $\bar{x}(t) := x(t) - x_{\text{nom}}(t)$ and $\bar{u}(t) := u(t) - u_{\text{nom}}(t)$:

$$\dot{\bar{x}}(t) = A(t)\bar{x}(t) + B(t)\bar{u}(t)$$

and associated **time-varying state feedback controller**,

$$\bar{u}(t) = -K(t)\bar{x}(t).$$

Derivation follows same lines as in previous time-invariant case:

$$J(\bar{x}', t') = \bar{x}^\top(t_f)Q_f\bar{x}(t_f) + \int_{t'}^{t_f} \left[\bar{x}^\top(t)Q\bar{x}(t) + \bar{u}^\top(t)R\bar{u}(t)\right] \, \mathrm{d}t$$

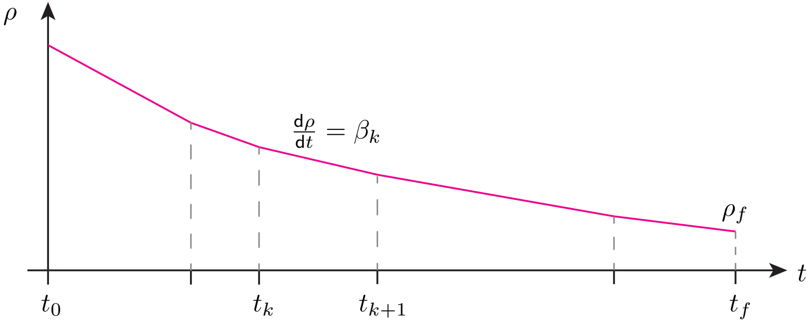
where $Q_f \succ 0$, $Q \succeq 0$, $R \succ 0$, and $\bar{x}(t') = \bar{x}'$. We have $K(t) = R^{-1}B^\top(t)S(t)$, where $S(t)$ solves the backward differential equation

$$-\dot{S} = Q - SBR^{-1}B^\top S + SA + A^\top S, \text{ with } S(t_f) = Q_f.$$

Closed-loop feedback along nonlinear trajectories (II)

- How far away from $(x_{\mathrm{nom}}(t), u_{\mathrm{nom}}(t))$ can we rely on $\bar{u}(t) = -K(t)\bar{x}(t)$?
  - The constant ρ found for the infinite-horizon problem earlier does not certify safety with time-varying dynamics.
- **Strategy:** Grow a time-varying funnel $\rho(t)$ **backwards from** $\rho(t_f) = \rho_f$
  - The end condition ρ_f is determined by whatever robustness is available at the part of the tree the nominal trajectory connects to at $t = t_f$
  - Then $\rho(t)$ is chosen to be piecewise linear on breakpoints in time $t_0, t_1, \ldots, t_k, \ldots, t_f$. These breakpoints often arise from the trajectory generation method.

Each segment has $\rho_k(t) = \alpha_k + \beta_k t$, with each α_k, β_k chosen such that $\rho_k(t_{k+1}) \leq \rho_{k+1}(t_{k+1})$.



Closed-loop feedback along nonlinear trajectories (III)

- Must ensure that $J(\bar{x}, t) := \bar{x}^\top(t)S(t)\bar{x}(t)$ decreases at least as fast as $\rho(t)$ whenever we are at the boundary of the funnel, i.e. whenever $J(\bar{x}, t) = \rho(t)$
- This is equivalent to

$$\dot{J}(\bar{x}, t) \leq \dot{\rho}_k(t) \equiv \beta_k, \quad \forall (\bar{x}, t) \in \{(\bar{x}, t) \,|\, J(\bar{x}, t) = \rho_k(t), t_k \leq t < t_{k+1}\}$$

  This is a polynomial non-negativity condition on a set defined by polynomial constraints!
- **SOS program**[3]

$$\begin{aligned}
\min_{h_1(\cdot), h_2(\cdot), h_3(\cdot)} \quad & 0 \\
\text{s. t.} \quad & \dot{J}(\bar{x}) - \dot{\rho}_k(t) + h_1(\bar{x}, t)(\rho_k(t) - J(\bar{x}, t)) \\
& \quad + h_2(\bar{x}, t)(t - t_k) + h_3(\bar{x}, t)(t_{k+1} - t) \leq 0, \quad \forall (\bar{x}, t) \\
& h_2 \in \Sigma[x]_{d_2}, \\
& h_3 \in \Sigma[x]_{d_3}.
\end{aligned}$$

[3] Note h_1 does not need to be non-negative because it is associated with an equality constraint, which is equivalent to two inequalities back-to-back $\Rightarrow$ sign of polynomial does not matter.

Implementing real-time control

- In real time we need to work out, given current state x, what branch b of the tree $\mathcal{T}$ we are on, and how far along the corresponding trajectory $x^b_{\text{nom}}(t)$.
- **Strategy:** compute a confidence score:

  $$c(x, t, b) := \rho^b(t) - (x - x^b_{\text{nom}}(t))^\top S^b(t)(x - x^b_{\text{nom}}(t))$$

  Intuitively this tells you how safely inside the funnel you are.
- Then our best guess of position is

  $$\underset{t \in [t_0, t_f], b \in \mathcal{T}}{\arg\max} \; c(x, t, b) \tag{C}$$

  and we implement the corresponding controller $K(t)$ from branch b and time t.
- In pratice, it is **not practical** to keep re-optimising b and t, so we just assume $\dot{t} = 1$ and that b only changes when the parent branch of the tree is reached.
  - At that point, $b \leftarrow \texttt{Parent}(b)$ and $t \leftarrow t_0$ again, and the controller is switched to the parent branch's control rule.
  - If a large disturbance blows us off the current branch, we can re-evaluate (C).

Probabilistic feedback coverage

- The LQR-Trees algorithm comes with a **probabilistic feedback coverage** guarantee.
  - Similar to the probabilistic completeness of RRT.
- Required assumptions:
  1. The sampling probability density is non-zero everywhere in $\mathcal{X}$
  2. At the goal x_G the linearized system is controllable
  3. The system is locally, exponentially stabilizable towards all trajectories obeying $\dot{x}(t) = f(x(t), u(t))$
  4. The motion planner has a non-zero chance of successfully connecting a new sampled point x_{new} to the existing tree.

Theorem (Tedrake et al. 2010)

Let $\mathcal{C}_\infty$ be the limiting coverage of the LQR-Trees algorithm, and let $\mathcal{R}(x_G)$ be the set of states from which there exists a piecewise-continuous control signal $u(t)$ such that the state asymptotically approaches x_G. Then if the assumptions above hold, $cl(\mathcal{C}_\infty) = cl(\mathcal{R}(x_G))$, where $cl(\cdot)$ indicates the closure of a possibly open set.

Numerical example: Pendulum swing-up

Tedrake et al. 2010, Section 5

- Pendulum with dynamics

$$I\ddot{\theta}(t) + b\dot{\theta}(t) + mgl\sin\theta(t) = \tau(t)$$

  where the states and inputs are

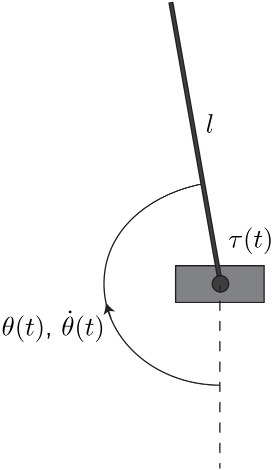
$$x(t) = \begin{bmatrix} \theta(t) \\ \dot{\theta}(t) \end{bmatrix}, \quad u(t) = \tau(t)$$

- Parameters: $m = 1$, $l = 0.5$, $b = 0.1$, $I = ml^2 = 0.25$, and $|\tau(t)| \le 3$.
- Goal is

$$x_G = \begin{bmatrix} \pi \\ 0 \end{bmatrix}, \quad u_G = 0$$

- LQR weightings:

$$Q = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}, \quad R = 20$$



Real-time control: Funnel libraries

Majumdar and Tedrake 2017

- A robot navigating in an unexplored environment cannot execute the LQR-Trees algorithm
- Furthermore, real-time computation of funnels is time consuming (SOS programs are potentially large semidefinite programs)
- **Strategy:** Pre-compute funnels for obstacle avoidance.
  - These funnels should be **as small as possible** to maximize chance of safe obstacle avoidance.
  - Determine whether offline computed funnels can be **composed in real time**.
  - In real-time, simply search for valid funnels once obstacles are observed.
- The result is a so-called **funnel library**, which can be queried in milliseconds, returning an associated feedback control law.
- Real-world hardware test on a small plane: `https://www.youtube.com/watch?v=cESFpLgSb50`

Notes and references

This lecture was based on the following publications:

Literature

S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

R. Tedrake *et al.*, "LQR-trees: Feedback Motion Planning via Sums-of-Squares Verification," *International Journal of Robotics Research*, vol. 29, no. 8, pp. 1038–1052, 2010.

A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *International Journal of Robotics Research*, vol. 38, no. 8, pp. 947–982, 2017.